

IDENTIFICATION

PRODUCT CODE: AH-T734A-DD
DIAGNOSTIC CODE: DFCIB
PRODUCT NAME: DFCIBB0 CI20 CTP Exerciser
VERSION: 2
DATE RELEASED: August 27, 1985
MAINTAINED BY: Diagnostic Engineering
AUTHOR: Bill Scorzelli
Ed Tulloch
UPDATE AUTHOR: Gregory A. Scott

COPYRIGHT (C) 1984, 1985

DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.

THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY ON A SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE, OR ANY OTHER COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE TERMS. TITLE TO AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DIGITAL EQUIPMENT CORPORATION.

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.

DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE IN EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL EQUIPMENT CORPORATION.

Table of Contents

Page

1.0	Abstract	1
2.0	Requirements	2
2.1	Hardware	2
2.2	Software	2
3.0	Loading/Starting Procedure	3
4.0	General Instructions	4
4.1	CTP Responder	4
4.2	CTP Exerciser	4
5.0	Commands	5
5.1	ACCEPT Command	5
5.2	BUILD Command	5
5.3	CLEAR Command	5
5.3.1	CLEAR LOGGING Command	6
5.3.2	CLEAR SWITCH Command	6
5.4	CONNECT Command	6
5.5	DDT Command	6
5.6	DESELECT Command	6
5.7	DISCONNECT Command	7
5.8	HELP Command	7
5.9	LISTEN Command	7
5.10	MAP-BUFFER Command	7
5.11	PAUSE Command	7
5.12	QUIT Command	7
5.13	READ Command	7
5.14	REQUEST-DATA Command	7
5.15	REQUEUE Command	8
5.16	RETURN-MESSAGE Command	8
5.17	RUN Command	8
5.18	SELECT Command	9
5.19	SET Command	9
5.19.1	SET LOGGING Command	9
5.19.2	SET PATH-SELECTION Command	9
5.19.3	SET SWITCH Command	9
5.19.4	SET TIME-OUT Command	9
5.19.5	SET TYPE-OUT-LINE-LIMIT Command	10
5.20	SHOW Command	10
5.20.1	SHOW ALL-PROGRAM-PARAMETERS Command	10
5.20.2	SHOW CI-CONFIGURATION Command	10
5.20.3	SHOW COUNTERS Command	10
5.20.4	SHOW EVENT-QUEUE Command	10
5.20.5	SHOW ID-OF-CONNECTED-NODE Command	11
5.20.6	SHOW LOCAL-NODE-NUMBER Command	11
5.20.7	SHOW MAPPED-BUFFER-NAME Command	11
5.20.8	SHOW MINIMUM-BUFFER-SIZES Command	11
5.20.9	SHOW PATH-SELECTION Command	11
5.20.10	SHOW PATH-STATUS-OF-NODE Command	11
5.20.11	SHOW POLL-STATUS-OF-CONNECTED-NODE Command	11

5.20.12	SHOW RUN-TABLE Command	12
5.20.13	SHOW SELECTED-NODES Command	12
5.20.14	SHOW STATUS-OF-CONNECTED-NODE Command	12
5.20.15	SHOW SWITCHES Command	12
5.20.16	SHOW TIME-OUT Command	12
5.21	TAKE Command	12
5.22	TYPE Command	13
5.23	WAIT Command	13
6.0	Test Descriptions	14
6.1	Error Detection	14
6.2	Overview of Tests	14
6.3	Basic Tests	15
6.3.1	TST01 Connect/Disconnect	15
6.3.2	TST02 Function Set Request/Response	15
6.4	Message Tests	15
6.4.1	TST10 Basic Message Test	15
6.4.2	TST11 Message, Count 4, Four Data Types	16
6.4.3	TST12 Message, Repeat 3, Four Data Types	16
6.4.4	TST13 Message, Delay 4, Four Data Types	17
6.4.5	TST14 Message, Size 1 to Maximum, Four Data Types	17
6.5	Datagram Tests	18
6.5.1	TST30 Basic Datagram Test	18
6.5.2	TST31 Datagram, Size 4, Four Data Types	18
6.5.3	TST32 Datagram, Repeat 3, Four Data Types	19
6.5.4	TST33 Datagram, Repeat and Delay 4, Four Data Types	19
6.5.5	TST40 Datagram Test Under Message Service	20
6.6	Move Buffer Tests	20
6.6.1	TST50 Buffer Map/Buffer Unmap	20
6.6.2	TST51 Buffer Map, Controller Read, Buffer Unmap	20
6.6.3	TST52 Buffer Map, Buffer Move, Buffer Unmap	21
6.6.4	TST53 Buffer Move (Delay 4)	22
6.6.5	TST54 Buffer Move (Repeat 3)	22
6.6.6	TST55 Buffer Move (Variable Length)	23
6.6.7	TST56 Buffer Move (Size 1)	24
6.6.8	TST57 Buffer Move (Multiple 1)	24
6.6.9	TST58 Buffer Move (Exerciser-Responder Write)	25
6.6.10	TST59 Buffer Move (Exerciser-Responder Write)	26
6.7	TST80 Read Counter Test	26
6.8	TST99 CTP Exerciser	27
7.0	Script Descriptions	28
7.1	DEFAULT Script	28
7.2	ALL-TESTS Script	28
7.3	BASIC-TESTS Script	28
7.4	BUFFER-TESTS Script	28
7.5	COUNTER-TEST Script	29
7.6	DATA-TESTS Script	29
7.7	EXERCISER Script	29
7.8	MESSAGE-TESTS Script	29

1.0 Abstract

The Computer Interconnect Cluster Test Protocol Exerciser (DFCIB) is designed to run in user mode only. It will exercise the entire CI cluster or a desired subset of it. It is designed to guarantee the integrity of the CI and isolate faults to a failing node.

DFCIB is to be the driver (controller) in a two process system. The driven process (responder) will be implemented by each node in the CI cluster. DFCIB will communicate with responders using the Cluster Test Protocol (CTP) and services provided under the Systems Communications Service (SCS). It ensures compliance with the CI Specification for System ID, Message, Datagram, and Data Transfer functions.

2.0 Requirements

2.1 Hardware

- o KL10 CPU
- o User or Console Terminal
- o Enough memory to support normal Timesharing Operating System
- o The load device is normally a system disk drive
- o Properly configured CI20 and Star Coupler
- o At least one other CI20 or HSC50 node on the CI

2.2 Software

- o TOPS-10 Operating System release 7.03 or later
- o TOPS-20 Operating System release 6.0 or later
- o DFCIB.EXE and DFCIB.HLP accessible on SYS: or DSK:
- o GLXLIB.EXE on SYS: (furnished with TOPS-10/TOPS-20)
- o Either the DFCIC CTP Responder running in another KL10
- o or the Responder software loaded in the HSC50

3.0 Loading/Starting Procedure

To load and start the exerciser do a 'RUN DFCIB', causing DFCIB.EXE to be loaded and executed. DFCIB.HLP must exist on DSK:, HLP:, or SYS: in order to access the on-line help functions.

GLXLIB.EXE will be loaded by DFCIB early in its initialization and must exist in SYS: in order to run DFCIB. GLXLIB will be accessible if TOPS-10/TOPS-20 is running since it is an integral part of operating system support utilities.

Under TOPS-10, the user must be running as [1,2] or have the JP.POK privilege. DFCIB locks itself in core in order to do buffer maps/unmaps while running a test.

Under TOPS-20, the user must have WHEEL or OPERATOR capability enabled.

4.0 General Instructions

DFCIB is run to cause traffic over the CI. It consists of two basic parts: the functional tests, and the CI exerciser. The functional tests (TST01-TST80) are used to verify the operation of the software and hardware that will be used in the exerciser (TST99). The exerciser is used to produce the heaviest traffic on the CI.

4.1 CTP Responder

In order to run the DFCIB Cluster Test Protocol Exerciser program on the CI, there must be at least one other node active running the Cluster Test Protocol Responder.

On HSC50 systems the responder software is activated with the "SET LOAD RS" command. This command will cause the responder software to be loaded at the next HSC50 reload. The HSC50 responder software may be left running with no ill effects.

On TOPS-10/20 systems, the DFCIC responder program must be running prior to running DFCIB. This is done by doing a "RUN DFCIC" command on an OPERATOR or [1,2] job. After DFCIB execution has finished, the DFCIC program should be stopped since it used system SCS resources (and under TOPS-10 it locks itself in core).

4.2 CTP Exerciser

To run DFCIB you should first select a node or nodes for testing (SELECT NODE n command). Then you will probably want to verify CTP operation by running the functional tests (RUN ALL-TESTS command). The most traffic is generated on the CI by TST99, usually invoked by the RUN EXERCISER command. The first time you run DFCIB you will probably want to SET SWITCH TRACE so that you can monitor the progress of the tests.

5.0 Commands

This section explains the commands available in DFCIB and how they are used.

Certain of the commands are for use by tech support personnel to write their own tests for situations not covered by the pre-written tests. This would allow them to:

- o Build CTP packets.
- o Send the packets.
- o Show the event queue and status to check the status of the packet that was sent and to wait for the receipt of the response message.
- o Read the response packet into the buffer.
- o Type the response message on the terminal.

The rest of the commands are used for normal operation of the exerciser program. Help for a specific command may be obtained with the HELP command as follows:

HELP command-name

For instance if you wish additional information about the TAKE command you would type 'HELP TAKE'.

5.1 ACCEPT Command

The ACCEPT command is used only for writing your own tests. It causes a connection to be accepted from another exerciser program.

5.2 BUILD Command

BUILD is a command used only for writing your own tests. It allows the building of CTP packets for later transmission using the SEND command.

5.3 CLEAR Command

The CLEAR command is the complement of the SET command.

5.3.1 CLEAR LOGGING Command

The CLEAR LOGGING command is used to close the log file, which is opened by the SET LOGGING command. The log file receives a copy of all output directed to the terminal.

5.3.2 CLEAR SWITCH Command

The CLEAR SWITCH command is used to clear one or more of the switches. It is the complement of the SET SWITCH command. The current switch settings are displayed with the SHOW SWITCHES command. The switches are:

BELL	Ring the terminal bell on errors.
HALT	Stop testing when any error is detected.
PALL	Print all bytes in buffer (TYPE command).
TRACE	Print trace messages as each test is executed.

5.4 CONNECT Command

The CONNECT command is used only for writing your own tests. It causes a connection to be made to the last node specified in the SELECT command.

5.5 DDT Command

The DDT command is used only for debugging the diagnostic. It causes execution to be transferred to DDT if it is loaded. If DDT is not loaded it attempts to merge in from SYS:VMDDT.EXE (TOPS-10) or SYS:UDD1.EXE (TOPS-20).

5.6 DESELECT Command

The DESELECT command is used to disable selection of a node for testing. It is followed by one or more decimal node numbers, separated by commas. To see what nodes are selected, use the SHOW SELECTED-NODES command.

5.7 DISCONNECT Command

The DISCONNECT command is used only for writing your own tests. It causes the connection created by the CONNECT command to be broken.

5.8 HELP Command

The 'HELP' command followed by question mark (?) will provide the user with a list of the entries in the online help file. HELP followed by the subject will provide the user with a short message on the selected subject. Help with no subject specified will provide the user with a help message on using the 'HELP' command.

5.9 LISTEN Command

The LISTEN command is used only for writing your own tests. It causes the program to listen for a CTP connection from any node.

5.10 MAP-BUFFER Command

The MAP-BUFFER command is used only for writing your own tests. It causes a map of an internal buffer of the number of bytes specified in the command.

5.11 PAUSE Command

The PAUSE command is normally used only for writing your own tests. It causes the program to sleep for the number of seconds furnished in the command.

5.12 QUIT Command

The QUIT command causes the diagnostic to exit. If a log file is open it is closed before exiting.

5.13 READ Command

The READ command is used only for writing your own tests. It is followed by a keyword, DATAGRAM or MESSAGE, indicating the read action.

5.14 REQUEST-DATA Command

The REQUEST-DATA command is normally used only for writing your own tests. It is used to request DMA transfer data from either a RESPONDER-NODE or an EXERCISER-NODE, as specified in the command.

5.15 REQUEUE Command

The REQUEUE command is normally used only for writing your own tests. It causes a buffer used by the READ command to be freed up for later use.

5.16 RETURN-MESSAGE Command

The RETURN-MESSAGE command is used only for writing your own tests. It causes the diagnostic to return a message as a responder program would.

5.17 RUN Command

The 'RUN' command is used to start the execution of a test or group of tests. The format of the RUN command is:

```
RUN /PASSES:r test/ITERATIONS:n, test/ITERATIONS:n
```

The optional /PASSES switch must be specified before any test name, and specifies how many passes of the test to run. The default number of passes if /PASSES isn't specified is 1.

A test name or test script come next, separated by commas. Each test may be execute a number of times using the optional /ITERATIONS switch. The default number of iterations is 1, except for the EXERCISER and TST99, for which the default iteration count is 100.

A RUN command with no arguments is used to rerun the last successfully entered RUN command. The tests are stored in the 'run table', which can be displayed using the SHOW RUN-TABLE command.

An example run command would be

```
DFCIB>RUN /PASSES:100 BASIC/ITER:10, MESSAGE/ITER:5, COUNTER
```

This would run 100 passes, where each pass would consist of 10 iterations of the Basic Connect/Disconnect Tests, 5 iterations of the Message Tests, followed by one iteration of the Counter Test.

During the execution of a test, you can get status of the test or abort testing by using the 'run time commands'. The run time command options are:

```
H   For this message  
S   To get current test status  
T   To toggle the TRACE switch  
esc Escape to abort testing
```

For more help with the tests, see the descriptions of the tests and scripts.

5.18 SELECT Command

The SELECT command is used to select a node for CTP testing. It is followed by node numbers separated by commas. The selected nodes are displayed with the SHOW SELECTED-NODES command.

5.19 SET Command

The SET command is used to activate various program options.

5.19.1 SET LOGGING Command

The SET LOGGING command is followed by a filename into which all terminal output is placed. The default filename is DSK:DFCIB.LOG. To close the log file, either exit the diagnostic with the QUIT command or use the CLEAR LOGGING command.

5.19.2 SET PATH-SELECTION Command

The SET PATH-SELECTION command is used to select either Path A, Path B, or automatic path selection for all CTP packets sent over the CI. The default is to use automatic path selection, which will let the monitor pick the path to use, causing CTP packets to be split more or less evenly across both paths.

5.19.3 SET SWITCH Command

The SET SWITCH command is used to set one or more of the switches. It is the complement of the CLEAR SWITCH command. The current switch settings are displayed with the SHOW SWITCHES command. The switches are listed along with the CLEAR SWITCH command.

5.19.4 SET TIME-OUT Command

The SET TIME-OUT command is followed by the maximum number of seconds to wait for any event to complete. The default time out is 5 seconds. The SHOW TIME-OUT command is used to display the current time out value.

5.19.5 SET TYPE-OUT-LINE-LIMIT Command

The SET TYPE-OUT-LINE-LIMIT command is used to set the number of lines that are displayed with the TYPE command. The TYPE command is only used for writing your own tests.

5.20 SHOW Command

The SHOW command is used to display various program settings.

5.20.1 SHOW ALL-PROGRAM-PARAMETERS Command

The SHOW ALL-PROGRAM-PARAMETERS command does the equivalent of the following SHOW commands:

- LOCAL-NODE-NUMBER
- PATH-SELECTION
- SELECTED-NODES
- TIME-OUT
- RUN-TABLE
- SWITCHES

The other SHOW commands have to do with writing your own test or getting status of this node or other nodes on the CI.

5.20.2 SHOW CI-CONFIGURATION Command

The SHOW CI-CONFIGURATION command will request connect data from all other nodes on the CI and display various node-specific data such as software type and versions, buffer sizes, and so on.

5.20.3 SHOW COUNTERS Command

The SHOW COUNTERS function will cause the CI port counters to be read and displayed for this node on the CI.

5.20.4 SHOW EVENT-QUEUE Command

The SHOW EVENT-QUEUE command is used only when writing your own tests and shows the status of the event queue for the connection created with the CONNECT or LISTEN command.

5.20.5 SHOW ID-OF-CONNECTED-NODE Command

The SHOW ID-OF-CONNECTED-NODE command is used only for writing your own tests and shows the connect ID of the node connected with the CONNECT or LISTEN command.

5.20.6 SHCW LOCAL-NODE-NUMBER Command

The SHOW LOCAL-NODE-NUMBER shows the CI node number assigned to the system the diagnostic is running on.

5.20.7 SHOW MAPPED-BUFFER-NAME Command

The SHOW MAPPED-BUFFER-NAME command is used only for writing your own tests and is used to display the buffer name returned by the MAP-BUFFER command.

5.20.8 SHOW MINIMUM-BUFFER-SIZES Command

The SHOW MINIMUM-BUFFER-SIZES command shows what monitor determines is the minimum buffer sizes for datagram and messages.

5.20.9 SHOW PATH-SELECTION Command

The SHOW PATH-SELECTION command will display what paths have been selected with the SET PATH-SELECTION command.

5.20.10 SHOW PATH-STATUS-OF-NODE Command

The SHOW PATH-STATUS-OF-NODE command, followed by a node number, shows what monitor determines is the current status of paths to that node.

5.20.11 SHOW POLL-STATUS-OF-CONNECTED-NODE Command

The SHOW POLL-STATUS-OF-CONNECTED-NODE command is used only for writing your own tests. It shows what monitor knows as the poll status of the currently connected node.

5.20.12 SHOW RUN-TABLE Command

The SHOW RUN-TABLE command is used to display what tests or scripts have been run with the last RUN command. This list of tests/scripts will be used if another RUN command is given with no tests/scripts specified. See the RUN command for more details.

5.20.13 SHOW SELECTED-NODES Command

The SHOW SELECTED-NODES command displays which nodes on the CI have been selected for testing. Nodes are selected for testing by using the SELECT NODE command and are removed from testing by the DESELECT NODE command.

5.20.14 SHOW STATUS-OF-CONNECTED-NODE Command

The SHOW STATUS-OF-CONNECTED-NODE command is used for writing your own tests. It displays the status of the connection made with the CONNECT or LISTEN commands.

5.20.15 SHOW SWITCHES Command

The SHOW SWITCH command is used to display the switch settings. The SET SWITCH command is used to set one or more of the switches. The CLEAR SWITCH command is used to clear switches. The switches are described along with the CLEAR SWITCH command.

5.20.16 SHOW TIME-OUT Command

The SHOW TIME-OUT command is used to display the time in seconds to wait for any event to complete. The default time out is 5 seconds. The SET TIME-OUT command is used to change the current time out value.

5.21 TAKE Command

The TAKE command is used to cause a file of commands to be passed to the program. The default filename is DSX:DFCIB.CMD. For example if the following commands were in a file called TEST15.CMD:

```
SELECT NODE 15  
SET LOGGING TEST15.LOG  
RUN/PASS:3 BUFFER-TESTS/ITER:10,COUNTER-TEST  
QUIT
```

The command TAKE TEST15 would cause node 15 to be selected, a log file

called TEST15.LOG be opened, three passes of the specified tests to be run, and the program to exit.

5.22 TYPE Command

The TYPE command is used for writing your own tests. It is followed by one of the following to display the contents of program buffers:

CTPPKT	DBUF1	DBUF2
DBUF3	DBUF4	DBUF5
DBUF6	MAPPED-BUFFER	MBUF1
MBUF2	MBUF3	MBUF4
MBUF5	MBUF6	RETURNED-BUFFER
SAVCTP	SAVREQ	SAVRSP
SAVSCS	SCSCMD	SCSEVT
SCSRSP	SCSSAV	

5.23 WAIT Command

The WAIT command used only for writing your own tests. It is used to wait for one of the following:

CONNECTION-STATUS STATE, one of the following:

ACCEPT-REQUEST-SENT	CLOSED
CONNECT-REQUEST-RECEIVED	CONNECT-REQUEST-SENT
CONNECT-RESPONSE-RECEIVED	CONNECTION-IS-OPEN
DISCONNECT-REQUEST-RECEIVED	DISCONNECT-REQUEST-SENT
DISCONNECT-RESPONSE-RECEIVED	LISTENING
REJECT-REQUEST-SENT	WAITING-FOR-DISCONNECT-RESPONSE

CONNECTION-STATUS FLAG, of one of the following:

DATAGRAM-AVAILABLE-FLAG	DMA-TRANSFER-COMPLETE
EVENT-PENDING-FLAG	MESSAGE-AVAILABLE-FLAG

EVENT-STATUS, one of the following:

CONNECT-TO-LISTEN	CONNECTION-WAS-ACCEPTED
CONNECTION-WAS-REJECTED	CREDIT-IS-AVAILABLE
LITTLE-CREDIT-LEFT	MAINT-DATA-XFER-COMPLETE
MESSAGE-DATAGRAM-SEND-COMPLETE	NODE-CAME-ONLINE
NODE-WENT-OFFLINE	OK-TO-SEND-DATA
PORT-BROKE-CONNECTION	REMOTE-INITIATED-DISCONNECT
VC-BROKEN	

6.0 Test Descriptions

This section will describe the tests and explain the limitations of each test. Testing is accomplished via passing CTP (Cluster Test Protocol) packets between the exerciser and the responder. The exerciser will send the commands (via request packets) to the responder which answers with response packets. The Systems Communications Services (SCS) of the operating system will be used to send CTP messages over the CI.

6.1 Error Detection

There are two types of errors that are detected:

- o A failure detected at the CTP level by the responder not responding properly, or
- o A failure at the SCS level, where the SCS or the operating system could not do the requested operation (SCS JSYS and SCS UUD).

There is no inherent fault isolation capability within the program. Any functional fault that is detected will be passed back to the exerciser for reporting to the user. Analysis and interpretation of the error will be up to the user. The exerciser will isolate to the node and identify the function being performed when the error occurred.

The exerciser will report errors to SPEAR under TOPS-20. DFCIB will also explain the commands available and how they are used.

6.2 Overview of Tests

DFCIB contains a number of tests, to test various parts of the CTP hardware/software unit, and the "exerciser" which is used to cause traffic loading over the CI.

To utilize the "exerciser" (TST99) portion of the exerciser, all previous tests should be run on the selected units. This is done to prove that all the CTP commands can be executed prior to using them in the "exerciser".

The test control code will run all functional tests on the first selected unit, then on to the next selected unit, and so on.

However, TST99 runs all of the nodes under test in parallel. This is done to increase traffic over the CI after all CTP functions have been verified.

Each test is explained below. The tests fall into seven categories:

- o Basic Tests: TST01, TST02
- o Message Tests: TST10, TST11, TST12, TST13, TST14
- o Datagram Tests: TST30, TST31, TST32, TST33, TST40
- o Buffer Tests: TST50, TST51, TST52, TST53, TST54, TST55, TST56, TST57, TST58, TST59
- o Counter Test: TST80
- o Exerciser Test: TST99

6.3 Basic Tests

6.3.1 TST01 Connect/Disconnect

This test will connect to the selected node and verify proper status (Event Pending, Connection Accepted, and Connection Open).

The test will then disconnect from the selected node and verify proper status (Connection Closed).

6.3.2 TST02 Function Set Request/Response

This test will connect to the selected node and verify proper status (Event Pending, Connection Accepted, and Connection Open).

A Function Set Request (opcode=0) will be sent and proper status verified (Event Pending, Message Send Complete, and Message Available). The Function Set Response message is read and verified (response opcode=64, status=0, and reference number equal to that sent).

The test will then disconnect from the selected node and verify proper status (Connection Closed).

6.4 Message Tests

6.4.1 TST10 Basic Message Test

This test will connect to the selected node and verify proper status (Event Pending, Connection Accepted, and Connection Open).

A Generate Message Request (opcode=4) with a word count of 0 will be sent and proper status verified (Event Pending, Message Send Complete, and Message Available). The Response message is read and verified (response=68, status=0, actual count=0, and reference number equal to that sent).

The test will then disconnect from the selected node and verify proper status (Connection Closed).

6.4.2 TST11 Message, Count 4, Four Data Types

This test will connect to the selected node and verify proper status (Event Pending, Connection Accepted, and Connection Open).

- (1) A Generate Message Request (opcode=4) with a word count of 4 and buffer fill data of 377 will be sent and proper status verified (Event Pending, Message Send Complete, and Message Available). The Response message is read and verified (response=68, status=0, actual count=0, and reference number equal to that sent). The data field is verified as 4 bytes of 377.
- (2) Identical to (1) except byte pair data is used for the fill data.
- (3) Identical to (1) except the responder node number is used for the fill data.
- (4) Identical to (1) except 4 bytes of image data (374, 375, 376, 377) is used for the fill data.

The test will then disconnect from the selected node and verify proper status (Connection Closed).

6.4.3 TST12 Message, Repeat 3, Four Data Types

This test will connect to the selected node and verify proper status (Event Pending, Connection Accepted, and Connection Open).

- (1) A Generate Message Request (opcode=4) with a word count of 4 and buffer fill data of 376 and a repeat count of 3 (which will generate 4 response messages) will be sent and proper status verified (Event Pending, Message Send Complete, and Message Available).

The 1st response message is read and verified (response=68, status=0, actual count=0, and reference number equal to that sent). The data field is verified as 4 bytes of 376.

The 2nd response message is read and verified (response=68, status=0, actual count=1, and reference number equal to that sent). The data field is verified as 4 bytes of 376.

The 3rd response message is read and verified (response=68, status=0, actual count=2, and reference number equal to that sent). The data field is verified as 4 bytes of 376.

The 4th response message is read and verified (response=68, status=0, actual count=3, and reference number equal to that sent). The data field is verified as 4 bytes of 376.

- (2) Identical to (1) except byte pair data is used for buffer fill data.
- (3) Identical to (1) except the responder node number is used for buffer fill data.
- (4) Identical to (1) except 4 bytes of image data (374, 375, 376, 377) is used for buffer fill data.

The test will then disconnect from the selected node and verify proper status (Connection Closed).

6.4.4 TST13 Message, Delay 4, Four Data Types

This test will connect to the selected node and verify proper status (Event Pending, Connection Accepted, and Connection Open).

- (1) A Generate Message Request (opcode=4) with a word count of 4 and buffer fill data of 375 and a delay count of 4 (2 seconds) will be sent and proper status verified (Event Pending, Message Send Complete, and Message Available). The response message is read and verified (response opcode=68, status=0, actual count=0, and reference number equal to that sent). The data field is verified as 4 bytes of 375.
- (2) Identical to (1) except byte pair data is used for buffer fill.
- (3) Identical to (1) except the responder node number is used for buffer fill.
- (4) Identical to (1) except 4 bytes of image data (374, 375, 376, 377) is used for buffer fill.

The test will then disconnect from the selected node and verify proper status (Connection Closed).

6.4.5 TST14 Message, Size 1 to Maximum, Four Data Types

This test will connect to the selected node and verify proper status (Event Pending, Connection Accepted, and Connection Open). Each of the following is done with message size varying from 1 byte to the maximum supported in the responder:

- (1) A Generate Message Request (opcode=4) with a variable byte count of 1 to maximum supported in the responder and buffer fill data of 373 will be sent and proper status verified (Event Pending, Message Send Complete, and Message Available). The Response messages are read and verified (response opcode=68, status=0, actual count=0, and reference number equal to that sent). The data field is verified as the correct number of bytes of 373.
- (2) Identical to (1) except byte pair data is used for buffer fill.

(3) Identical to (1) except the responder node number is used for buffer fill.

(4) Identical to (1) except 4 bytes of image data (374, 375, 376, 377) is used for buffer fill.

The test will then disconnect from the selected node and verify proper status (Connection Closed).

6.5 Datagram Tests

6.5.1 TST30 Basic Datagram Test

This test will connect to the selected node and verify proper status (Event Pending, Connection Accepted, and Connection Open). A Generate Datagram Request (opcode=5) with a word count of 0 will be sent and proper status verified (Event Pending, Message Send Complete, and Datagram Available). The Response Datagram is read and verified (response opcode=69, status=0, actual count=0, and reference number equal to that sent). The test will then disconnect from the selected node and verify proper status (Connection Closed).

6.5.2 TST31 Datagram, Size 4, Four Data Types

This test will connect to the selected node and verify proper status (Event Pending, Connection Accepted, and Connection Open).

(1) A Generate Datagram Request (opcode=5) with a word count of 4 and buffer fill data of 377 will be sent and proper status verified (Event Pending, Message Send Complete, and Datagram Available). The Response Datagram is read and verified (response opcode=69, status=0, actual count=0, and reference number equal to that sent). The data field is verified as 4 bytes of 377.

(2) Identical to (1) except byte pair data is used for buffer fill.

(3) Identical to (1) except the responder node number is used for buffer fill.

(4) Identical to (1) except 4 bytes of image data (374, 375, 376, 377) is used for buffer fill.

The test will then disconnect from the selected node and verify proper status (Connection Closed).

6.5.3 TST32 Datagram, Repeat 3, Four Data Types

This test will connect to the selected node and verify proper status (Event Pending, Connection Accepted, and Connection Open).

- (1) A Generate Datagram Request (opcode=5) with a byte count of 4 and buffer fill data of 376 and a repeat count of 3 (which will generate 4 response datagrams) will be sent and proper status verified (Event Pending, Message Send Complete, and Datagram Available).

The 1st Response Datagram is read and verified (response opcode=69, status=0, actual count=0, and reference number equal to that sent). The data field is verified as 4 bytes of 376.

The 2nd Response Datagram is read and verified (response opcode=69, status=0, actual count=0, and reference number equal to that sent). The data field is verified as 4 bytes of 376.

The 3rd Response Datagram is read and verified (response opcode=69, status=0, actual count=0, and reference number equal to that sent). The data field is verified as 4 bytes of 376.

The 4th Response Datagram is read and verified (response opcode=69, status=0, actual count=0, and reference number equal to that sent). The data field is verified as 4 bytes of 376.

- (2) Identical to (1) except byte pair data is used.
- (3) Identical to (1) except responder node number is used for the data.
- (4) Identical to (1) except image data of 374, 375, 376, 377 is used.

The test will then disconnect from the selected node and verify proper status (Connection Closed).

6.5.4 TST33 Datagram, Repeat and Delay 4, Four Data Types

This test will connect to the selected node and verify proper status (Event Pending, Connection Accepted, and Connection Open).

- (1) A Generate Datagram Request (opcode=5) with a word count of 4 and buffer fill data of 375 and a delay count of 4 (2 seconds) will be sent and proper status verified (Event Pending, Message Send Complete, and Datagram Available). The Response Datagram is read and verified (response opcode=69, status=0, actual count=0, and reference number equal to that sent). The data field is verified as 4 bytes of 375.

- (2) Identical to (1) except byte pair data is used.
- (3) Identical to (1) except responder node number is used as data.
- (4) Identical to (1) except image data of 374, 375, 376, 377 is used.

The test will then disconnect from the selected node and verify proper status (Connection Closed).

6.5.5 TST40 Datagram Test Under Message Service

This test will connect to the selected node and verify proper status (Event Pending, Connection Accepted, and Connection Open).

A Generate Datagram Request (opcode=5) under Message Service with a word count of 26, buffer fill data of 201, a delay count of 4 (2 seconds), and a repeat count of 1 will be sent and proper status verified (Event Pending, Message Send Complete, and Datagram Available). The Response Datagram is read and verified (response opcode=69, status=0, actual count=0, and reference number equal to that sent). The data field is verified as 4 bytes of 201.

The ending response message is read and verified (response opcode=69, status=0, actual count=1, and reference number equal to that sent). The data field is verified as 4 bytes of 201.

The test will then disconnect from the selected node and verify proper status (Connection Closed).

6.6 Move Buffer Tests

6.6.1 TST50 Buffer Map/Buffer Unmap

This test will connect to the selected node and verify proper status (Event Pending, Connection Accepted, and Connection Open).

A Buffer Map Request (opcode=1) for a real buffer with a buffer fill of 377 and a buffer length of 576 bytes is sent and proper status verified (Event Pending, Message Send Complete, and Message Available). The response message is read and verified (response opcode=65, status=1).

A Buffer Unmap Request (opcode=2) for the same buffer is sent and proper status is verified (Event Pending, Message Send Complete, and Message Available). The Response Message is read and verified (response opcode=66, status=1).

6.6.2 TST51 Buffer Map, Controller Read, Buffer Unmap

This test will not run against an HSC50 responder. If the responder software type is HSC, the test will return immediately. If the node under test is not an HSC responder, this test will connect to the selected node and verify proper status (Event Pending, Connection Accepted, and Connection Open).

(1) A Buffer Map Request (opcode=1) for a real buffer with a buffer fill

of 252 and a buffer length of 4 bytes is sent and proper status verified (Event Pending, Message Send Complete, and Message Available). The Response Message is read and verified (response opcode=65, status=1).

An internal buffer is mapped (in the exerciser), and bytes of 125 are written into the buffer.

Data is read, and the buffer is verified to contain bytes of 252.

The internal buffer is unmapped.

A Buffer Unmap Request (opcode=2) for the responder buffer is sent and proper status is verified (Event Pending, Message Send Complete, and Message Available). The Response Message is read and verified (response opcode=66, status=1).

(2) Identical to (1) except that the data is byte pairs.

(3) Identical to (1) except that the data is responder node number.

The test will then disconnect from the selected node and verify proper status (Connection Closed).

6.6.3 TST52 Buffer Map, Buffer Move, Buffer Unmap

This test will connect to the selected node and verify proper status (Event Pending, Connection Accepted, and Connection Open).

A Buffer Map Request (opcode=1) for a real buffer with a buffer fill of 252 and a buffer length of 4 bytes is sent and proper status verified (Event Pending, Message Send Complete, and Message Available). The Response Message is read and verified (response opcode=65, status=1).

An internal buffer is mapped and bytes of 125 are written into the buffer.

A Move Buffer Request (opcode=3) is issued for a responder write of 4 bytes with a delay, a repeat count, a packet multiplier and a packet size of 0 is sent and proper status verified (Event Pending, Message Send Complete, and Message Available). The Move Buffer Response Message is read and verified (response opcode=67, status=1).

The buffer is verified to contain bytes of 252.

The internal buffer is unmapped.

A Buffer Unmap Request (opcode=2) for the same buffer is sent and proper status is verified (Event Pending, Message Send Complete, and Message Available). The Response Message is read and verified (response opcode=66, status=1).

The test will then disconnect from the selected node and verify proper status (Connection Closed).

6.6.4 TST53 Buffer Move (Delay 4)

This test will connect to the selected node and verify proper status (Event Pending, Connection Accepted, and Connection Open).

A Buffer Map Request (opcode=1) for a real buffer with a buffer fill of 371 and a buffer length of 4 bytes is sent and proper status verified (Event Pending, Message Send Complete, and Message Available). The Response Message is read and verified (response opcode=65, status=1).

An internal buffer is mapped and bytes of 125 are written into the buffer.

A Move Buffer Request (opcode=3) is issued for a responder write of 4 bytes with a repeat count, a packet multiplier and a packet size of 0 and a delay count of 4 is sent and proper status verified (Event Pending, Message Send Complete, and Message Available). The Move Buffer Response Message is read and verified (response opcode=67, status=1).

The buffer is verified to contain bytes of 371.

A Buffer Unmap Request (opcode=2) for the same buffer is sent and proper status is verified (Event Pending, Message Send Complete, and Message Available). The response message is read and verified (response opcode=66, status=1).

The internal buffer is unmapped.

The test will then disconnect from the selected node and verify proper status (Connection Closed).

6.6.5 TST54 Buffer Move (Repeat 3)

This test will connect to the selected node and verify proper status (Event Pending, Connection Accepted, and Connection Open).

A Buffer Map Request (opcode=1) for a real buffer with a buffer fill of 370 and a buffer length of 4 bytes is sent and proper status verified (Event Pending, Message Send Complete, and Message Available). The Response Message is read and verified (response opcode=65, status=1).

An internal buffer is mapped and filled with bytes of 125.

A Move Buffer is issued for a responder write of 4 bytes with a delay, a packet multiplier and a packet size of 0 and a repeat count of 3 is sent and proper status verified (Event Pending, Message Send Complete, and Message Available). The Response Message is read and verified (response opcode=65, status=1, actual count=3).

The buffer contents is verified to contain bytes of 370.

A Buffer Unmap Request (opcode=2) for the same buffer is sent and proper status is verified (Event Pending, Message Send Complete, and Message Available). The response message is read and verified (response opcode=66, status=1).

The internal buffer is unmapped.

The test will then disconnect from the selected node and verify proper status (Connection Closed).

6.6.6 TST55 Buffer Move (Variable Length)

This test will connect to the selected node and verify proper status (Event Pending, Connection Accepted, and Connection Open).

Do the following starting with a buffer length of 4 and incrementing the buffer length by four until a buffer length of 1024 is reached.

A Buffer Map Request (opcode=1) for a real buffer with a buffer fill of byte pair and a buffer length of 1024 bytes is sent and proper status verified (Event Pending, Message Send Complete, and Message Available). The Response Message is read and verified (response opcode=65, status=1).

Map an internal buffer 1024 bytes in length fill it with bytes of 125.

A Move Buffer Request (opcode=3) for a responder write of variable length bytes with a delay, a repeat count, a packet multiplier and packet size of 0. Proper status verified (Event Pending, Message Send Complete, and Message Available). The Move Buffer Response Message is read and verified (response opcode=67, status=1).

Verify the buffer contains byte pair data.

A Buffer Unmap Request (opcode=2) for the same buffer is sent and proper status is verified (Event Pending, Message Send Complete, and Message Available). The response message is read and verified (response opcode=66, status=1).

The internal buffer is unmapped.

The test will then disconnect from the selected node and verify proper status (Connection Closed).

6.6.7 TST56 Buffer Move (Size 1)

This test will connect to the selected node and verify proper status (Event Pending, Connection Accepted, and Connection Open).

A Buffer Map Request (opcode=1) for a real buffer with a buffer fill of 366, a buffer length of 1024 bytes and a packet size of 1 is sent and proper status verified (Event Pending, Message Send Complete, and Message Available). The response message is read and verified (response opcode=65, status=1).

An internal buffer is mapped and filled with bytes of 125.

A Move Buffer Request (opcode=3) for a responder write of 1024 bytes with a delay, a repeat count, a packet multiplier of 0 and a packet size of 1 is sent and proper status verified (Event Pending, Message Send Complete, and Message Available). The Move Buffer Response Message is read and verified (response opcode=67, status=1).

The buffer is verified to contain 366.

A Buffer Unmap Request (opcode=2) for the same buffer is sent and proper status is verified (Event Pending, Message Send Complete, and Message Available). The response message is read and verified (response opcode=66, status=1).

The Internal buffer is unmapped.

The test will then disconnect from the selected node and verify proper status (Connection Closed).

6.6.8 TST57 Buffer Move (Multiple 1)

This test will connect to the selected node and verify proper status (Event Pending, Connection Accepted, and Connection Open).

A Buffer Map Request (opcode=1) for a real buffer with a buffer fill of 371 and a buffer length of 1024 bytes and a packet multiple of 1 is sent and proper status verified (Event Pending, Message Send Complete, and Message Available). The response message is read and verified (response opcode=65, status=1).

An internal buffer is mapped and filled with bytes of 125.

A Move Buffer Request (opcode=3) for a responder write of 1024 bytes with a delay count of 4, a repeat count of 0, a packet size of 0 and a packet multiplier of 1 is sent and proper status verified (Event Pending, Message Send Complete, and Message Available). The Move Buffer Response Message is read and verified (response opcode=67, status=1).

The buffer is verified to contain 252.

A Buffer Unmap Request (opcode=2) for the same buffer is sent and proper status is verified (Event Pending, Message Send Complete, and Message Available). The response message is read and verified (response opcode=66, status=1).

The internal buffer is unmapped.

The test will then disconnect from the selected node and verify proper status (Connection Closed).

6.6.9 TST58 Buffer Move (Exerciser-Responder Write)

This test runs only against the HSC responder. This test will connect to the selected node and verify proper status (Event Pending, Connection Accepted, and Connection Open).

A Buffer Map Request (opcode=1) for a real buffer with a buffer fill of 252 and a buffer length of 1024 bytes is sent and proper status verified (Event Pending, Message Send Complete, and Message Available). The Response Message is read and verified (response opcode=65, status=1).

Map first internal buffer in the exerciser and fill with bytes of 125.

A Move Buffer Request (opcode=3) for an exerciser write of 1024 bytes with a delay, a repeat count, a packet multiplier and a packet size of 0 is sent. Proper status is verified (Event Pending, Message Send Complete, and Message Available). The Response Message is read and verified (response opcode=65, status=1).

The first internal buffer is unmapped.

Map a second internal buffer and fill with 0 bytes.

A Move Buffer Request (opcode=3) for an responder write of 1024 bytes with a delay, a repeat count, a packet multiplier and a packet size of 0 is issued. Proper Status is verified (Event Pending, Message Send Complete, and Message Available). The Response Message is read and verified (response opcode=65, status=1).

The buffer is verified to contain bytes of 125.

A Buffer Unmap Request (opcode=2) for the same buffer is sent and proper status is verified (Event Pending, Message Send Complete, and Message Available). The response message is read and verified (response opcode=66, status=1).

Unmap the second internal buffer.

The test will then disconnect from the selected node and verify proper status (Connection Closed).

6.6.10 TST59 Buffer Move (Exerciser-Responder Write)

This test will not run against the HSC responder. This test will connect to the selected node and verify proper status (Event Pending, Connection Accepted, and Connection Open).

A Buffer Map Request (opcode=1) for a real buffer with a buffer fill of 252 and a buffer length of 1024 bytes is sent and proper status verified (Event Pending, Message Send Complete, and Message Available). The Response Message is read and verified (response opcode=65, status=1).

Map an internal buffer of 1024 bytes and fill with bytes of 125.

A Move Buffer Request (opcode=3) is sent to transfer the data to the responder from the exerciser. Test for data transfer complete.

Unmap the first internal buffer.

Map a second internal buffer fill with 0 bytes.

A Move Buffer Request (opcode=3) for a responder write of 1024 bytes with a delay, a repeat count, a packet multiplier and a packet size of 0 is sent and proper status verified (Event Pending, Message Send Complete, and Message Available). The response message is read and verified (response opcode=65, status=1).

Verify the buffer contains bytes of 125.

A Buffer Unmap Request (opcode=2) is sent and proper status is verified (Event Pending, Message Send Complete, and Message Available). The Response Message is read and verified (response opcode=66, status=1).

The test will then disconnect from the selected node and verify proper status (Connection Closed).

6.7 TST80 Read Counter Test

This test will connect to the selected node and verify proper status (Event Pending, Connection Accepted, and Connection Open).

A Counter Read Request (opcode=10) is issued and proper status is verified (Event Pending, Message Send Complete, and Message Available). The Counter Read Response Message is read and verified (response opcode=74, status=0).

The number of acks, nacks and no responses for path A and path B is printed out along with the number of disregarded datagrams.

The test will then disconnect from the selected node and verify proper status (Connection Closed).

6.8 TST99 CTP Exerciser

This test runs 3 iterations of the exerciser sequence while varying the repeat counts to 2, 32, and 64. This is done to increase the amount of traffic on the CI. The exerciser sequence utilizes the three basic methods of transfers (1) messages, (2) datagrams, and (3) buffer transfers, to create traffic on the CI.

The test begins by initializing the test tables and flags and goes on to get configuration data on all nodes on the CI. It uses this configuration data and the selected nodes to create selection list for the exerciser. Nodes that are selected but not accessible are dropped.

The exerciser sequence consists of the following for each node. All operations are performed on each node before moving to the next item in the list:

- o Connect to node.
- o Issue Generate Message Commands of 64 bytes.
- o Wait for and verify the Generate Message Responses.
- o Issue Generate Datagram Requests of 20 bytes.
- o Wait for and verify Generate Datagram Responses.
- o Map internal buffer, Issue Map Buffer Request for 64 bytes, verify Map Buffer Response.
- o Issue Move Buffer Request, wait for and verify Move Buffer Response.
- o Verify move buffer data transfers.
- o Issue Unmap Buffer Request, verify Unmap Buffer Response, Unmap internal buffer.
- o Disconnect from node.

Unlike the other tests, the default iteration count for TST99 is set to 100. Also unlike the other tests, if the TRACE switch is set a message is printed after each 10 iterations of the test.

7.0 Script Descriptions

7.1 DEFAULT Script

The DEFAULT script runs the same tests as the ALL-TESTS script except it selects all nodes available for testing first.

7.2 ALL-TESTS Script

The ALL-TESTS script runs of the following tests:

- TST01 Connect/Disconnect Test
- TST02 Function Set Request/Response Test
- TST10 Basic Message Test
- TST11 Message Test Count 4, Four Data Types
- TST12 Message Test Repeat 3, Four Data Types
- TST13 Message Test Delay 4, Four Data Types
- TST14 Message Test Size 1 to Maximum, Four Data Types
- TST30 Basic Datagram Test
- TST31 Datagram Test Size 4, Four Data Types
- TST32 Datagram Test Repeat 3, Four Data Types
- TST33 Datagram Repeat and Delay 4, Four Data Types
- TST40 Datagram Test Under Message Service
- TST50 Buffer Map and Buffer Unmap Test
- TST51 Buffer Map, Controller Read, Buffer Unmap
- TST52 Buffer Map, Buffer Move, Buffer Unmap
- TST53 Buffer Move (Delay 4)
- TST54 Buffer Move (Repeat 3)
- TST55 Buffer Move (Variable Length)
- TST56 Buffer Move (Size 1)
- TST57 Buffer Move (Multiple 1)
- TST58 Buffer Move (Exerciser-Responder Write) HSC50
- TST59 Buffer Move (Exerciser-Responder Write) TOPS-10/20/VMS
- TST80 Read Counter Test

7.3 BASIC-TESTS Script

The BASIC-TESTS script runs the following tests:

- TST01 Connect/Disconnect Test
- TST02 Function Set Request/Response Test

7.4 BUFFER-TESTS Script

The BUFFER-TESTS script runs the following tests:

- TST50 Buffer Map and Buffer Unmap Test
- TST51 Buffer Map, Controller Read, Buffer Unmap
- TST52 Buffer Map, Buffer Move, Buffer Unmap

TST53 Buffer Move (Delay 4)
TST54 Buffer Move (Repeat 3)
TST55 Buffer Move (Variable Length)
TST56 Buffer Move (Size 1)
TST57 Buffer Move (Multiple 1)
TST58 Buffer Move (Exerciser-Responder Write) HSC50
TST59 Buffer Move (Exerciser-Responder Write) TOPS-10/20/VMS

7.5 COUNTER-TEST Script

The COUNTER-TEST script just runs TST80, the Read Counter Test.

7.6 DATA-TESTS Script

The DATA-TESTS script runs the following tests:

TST30 Basic Datagram Test
TST31 Datagram Test Size 4, Four Data Types
TST32 Datagram Test Repeat 3, Four Data Types
TST33 Datagram Repeat and Delay 4, Four Data Types
TST40 Datagram Test Under Message Service

7.7 EXERCISER Script

The EXERCISER script is used to run TST99 only. Unlike the other scripts, the default iteration count is set to 100. It is suggested that at least 100 iterations of the exerciser be performed in order to verify proper operation over the CI.

7.8 MESSAGE-TESTS Script

The MESSAGE-TESTS script runs the following tests:

TST10 Basic Message Test
TST11 Message Test Count 4, Four Data Types
TST12 Message Test Repeat 3, Four Data Types
TST13 Message Test Delay 4, Four Data Types
TST14 Message Test Size 1 to Maximum, Four Data Types

[End of DFCIB.TXT]

Cluster Test Protocol (CTP)
for the
Computer Interconnect (CI)
Version 3
Revision 0

Revision History

Version 0
Revision 0

28 Jul 81

David McMillen TW/E05

Version 1
Revision 1
29 Mar 82
Jim Klumpp TW/F17

Version 2
Revision 0
24 Aug 82
Fred Roemer TW/F17

Version 3
Revision 0
12 Apr 83
Fred Roemer TW/F17

CONTENTS	PAGE
1.0 SUMMARY	3
2.0 RELATIONSHIPS AND LAYERING	4
3.0 DIAGNOSTIC PROCESSES	5
4.0 CTP FUNCTIONALITY	6
4.1 Executable Function Inquiry	6
4.2 Remote Buffer Map/Unmap	7
4.3 Remote Map/Unmap Maintenance Buffer	7
4.4 Generate Maintenance Packet	7
4.5 Perform Buffer Transfer	8
4.6 Generate Datagrams/Messages	8
4.7 Start / Stop Unsolicited Activity	8
4.8 Report Statistics/Status	8
5.0 CTP OPERATION	9
5.1 CTP Controller Operation	9
5.2 CTP Responder Operation	17
6.0 REQUIRED/OPTIONAL CTP REQUESTS	21
7.0 CTP MESSAGE FORMATS	23
7.1 Responder Function Set Request Message	23
7.2 Responder Function Set Response Message	23
7.3 Buffer Map Request Message	24
7.4 Buffer Map Response Message	25
7.5 Buffer Unmap Request Message	25
7.6 Buffer Unmap Response Message	26
7.7 Move Buffer Request Message	27
7.8 Move Buffer Response Message	27
7.9 Generate Message Request Message	28
7.10 Generate Message Response Message	29
7.11 Generate Datagram Request Message	30
7.12 Generate Datagram Response Message	30
7.13 Generate Reset Request Message	31
7.14 Generate Reset Response Message	31
7.15 Generate Start Request Message	32
7.16 Generate Start Response Message	32
7.17 Start / Stop Unsolicited Activity Request Message	33
7.18 Start / Stop Unsolicited Activity Response Message	34
7.19 Configuration Data Request Message	34
7.20 Configuration Data Response Message	34
7.21 Counter Read Request Message	35
7.22 Counter Read Response Message	35
7.23 Connect Request Message	36
7.24 Connect Response Message	36
7.25 Listener Disconnect Request Message	37
7.26 Listener Disconnect Response Message	37
7.27 Maintenance Buffer Map Request Message	37
7.28 Maintenance Buffer Map Response Message	38
7.29 Maintenance Buffer Unmap Request Message	38
7.30 Maintenance Buffer Unmap Response Message	39
7.31 Move Maintenance Buffer Request Message	39
7.32 Move Maintenance Buffer Response Message	40
7.33 Maintenance State Request Message	41
7.34 Maintenance State Response Message	41

APPENDIX A MULTIPLE CI DROPS ON ONE PROCESSOR

APPENDIX B MULTIPLE CTP CONTROLLERS ACTIVE ON ONE CI

APPENDIX C CLUSTER TEST SCENARIOS

C.1	BASIC SEQUENCED MESSAGE TEST	46
C.2	UUT READ FROM CINT BUFFER TEST.	47

APPENDIX D MESSAGE OFFSET SUMMARY

APPENDIX E OPCODE SUMMARY

APPENDIX F CTP STATUS AND VALUES SUMMARY

1.0 SUMMARY

This document describes the methods and protocols to be used for CI nodes to communicate when performing CI cluster diagnostic and test functions. The goals addressed are:

1. Cluster fault detection and isolation:

Provide a mechanism for diagnostic processes to control the use of the hardware in such a way as to isolate failed components within the cluster. This mechanism must allow for any node to control the tests, and condition other nodes to respond in the appropriate manner.

2. Long-term stability:

Since some nodes may desire to put the fundamental diagnostic and test communications software in ROM, the mechanisms used by an implementation should be stable over long periods of time.

There are no assumptions made about the internal architecture of any CI node using these facilities. In particular, this document intends for the cluster diagnostic and test activities of dissimilar nodes to be coordinated.

The functionality provided for in this document is specific to the CI, since it is used to test conformance to the CI architecture.

2.0 RELATIONSHIPS AND LAYERING

The CI is normally under the control of the Systems Communication Architecture (SCA). This architecture is specified in a document available from William Strecker (TW/B05).

The processes implementing this Cluster Test Protocol will reside at the SYSAP level described in the SCA. This places the CTP at the applications layer. Certain procedures will require management of other layers within the same node.

The diagnostic process will use the SCA as one of the tools to accomplish specific tests of the CI. The only other tool that is available is this test protocol. It is the sum of SCA and CTP that allow the cluster to be diagnosed. Neither alone will provide all functionality required.

3.0 DIAGNOSTIC PROCESSES

There are two projects currently underway as part of the IPA780 project. These are the CI Node Tester (CINT) and the CI Exerciser (CIE). Both will initially use the IPA780, with the CINT having a modified version of the hardware. The CIE is designed to operate on any VAX CI port. This protocol is expected to serve the needs of these projects, and their plans should be read for a better understanding of this protocol's intentions.

Furthermore, the HSC50 and Jupiter projects will add more CI node types. This protocol addresses the standardization needed for these varied nodes to jointly identify cluster problems. It is expected that the Jupiter project will produce something similar to the VAX CIE, while the HSC50 project will include the minimum functionality needed to participate.

4.0 CTP FUNCTIONALITY

There are two "sides" to the cluster test protocol. One is the "responder", responsible for recognizing an incoming request and responding in the correct manner. The other is the "controller", initiating a request and receiving the response.

Clearly there must be at least one node on each CI that is capable of driving the controller portion of the cluster test protocol. It is considered an invalid CI configuration if none of the nodes has this capability, since cluster failures could not be diagnosed. Therefore, what might be considered "root" nodes, such as VMS or TOPS10/20 systems, should implement the controller function.

All nodes must implement the responder portion of the cluster test protocol. This is a required vehicle for cluster fault analysis.

This protocol is designed to allow fault detection and isolation within the cluster. The functions do not support checkout of the basic CI interface. It is assumed that stand-alone diagnostics will diagnose the raw hardware at each node, in a manner that does not affect normal CI operation between other nodes. The purpose of these functions is to validate connection paths, check for compliance with the CI specification, and place test traffic levels on the CI.

The role of the responder is as a simple slave. Messages arrive with action requests, and the responder does as directed.

The role of the controller is as a master. The use of CTP in this role is straightforward, issuing action requests and getting responses. However, the controller will be building upon functions to perform a systematic checkout of the cluster and the abilities of individual nodes. It is the responsibility of the controller to perform tests, especially destructive ones, only when the system(s) being tested are in the appropriate state. This resource management is normally done at higher levels, invoking the controller code only when it can perform the tests requested. However, the responder may refuse to perform functions known to be harmful at that moment in time, and the controller may determine the current set of executable functions within any responder.

The following sections detail the functionality available:

4.1 Executable Function Inquiry

The responding system will indicate the set of opcodes that are implemented by the responder at that point in time. There is no broadcast upon a change in this function set, so the controllers must poll for any changes. This function allows a controller to determine the appropriate set of tests that may be run with a particular responder.

4.2 Remote Buffer Map/Unmap

The responding system will allocate or deallocate buffer memory and map or unmap it for access through the CI port that received the request. There are two types of buffers desired. One is a normal, real memory buffer, used to verify the data transfer mechanisms. The other is a "black hole", or a buffer that only appears to have real memory behind it. Black hole buffers are used only to generate data traffic on the CI, and the contents of the data will never be examined. The distinction is made to allow systems to optimize the resources given to the responder. If no optimizations can be made, normal buffers may be used to fill any requests for black hole buffers. It is possible for the responder to decline a request if the resources are not available. The minimum resources that the responder must make available are:

Real buffer space = 2048 bytes or 2 times the nodes largest possible CI data packet.

Black Hole buffer space = 16K Bytes.

Mapping descriptors = 2 for each controller connected to the responder.

These resources need only be available when the node is idle. This request can always be declined if the responding system does not support any buffer mode transfers.

4.3 Remote map/Unmap Maintenance Buffer

The responding system will allocate or deallocate a real memory buffer, and respond with the system-specific 32-bit identifier that will allow access. This request may be declined if resources are not available, subject to the same restrictions as the preceeding section. In addition, the size of the buffer requested may be reduced to a minimum of 512 bytes of contiguous space if the size requested cannot be allocated. This request can always be declined if the responding system does not support incoming maintenance read or write functions.

4.4 Generate Maintenance Packet

The responding system will generate the requested packet, with a requested target node. The contents of the packet will be specified in terms of the SCA maintenance arguments. There will be a repetition count and a zero to n second delay (in 500ms increments) prior to transmission request.

4.5 Perform Buffer Transfer

The responding system will perform a buffer transfer, with the transfer direction specified in the request. The name of the responder's buffer will be supplied in the request, along with the node address and buffer name to transfer to or from. There will be a repetition count and a zero to n second delay (in 500ms increments) prior to transmission request.

4.6 Generate Datagrams/Messages

The responding system will return to the sending node a datagram or message with the data field contents as specified in the request. The datagram or message will be the size of the data plus the CTP overhead in length. Messages will not exceed a maximum of 128 bytes total. There will be a repetition count and a zero to n second delay (in 500ms increments) prior to transmission request.

4.7 Start / Stop Unsolicited Activity

The responding system will either stop or resume unsolicited activity on the CI. For example, this means that if a node has a poller that checks configuration information by executing request ID's to other nodes, then upon receipt of this request this activity is stopped. This includes any CI activity that is originated within the node. This does not mean that a node does not respond to CTP requests, just that a node must not initiate unsolicited activity. This function is a CINT specific function. In the CINT configuration the poller will be controlled through the CI780 port, not the CINT port, so when the virtual circuit crashes between the CINT port and UUT port, the poller will be restarted using the virtual circuit between the CI780 port and UUT port.

4.8 Report Statistics/Status

The responding system will return the current statistics for the CI port that the request was received on, along with port status and information that SCA makes available.

5.0 CTP OPERATION

This section describes the manner in which the controller and responder establish communication, perform tests, and stop communication. The issues are addressed from the perspective of the controller. Only the subjects related to CTP are included here, as the greater goals of test and diagnosis belong to whoever implements the controller.

The second sub-section addresses the issues from the perspective of the responder.

5.1 CTP Controller Operation

The CTP controller is a trusted system process with the ability to examine and control the local SCA processes, as well as initiating and operating SCA connections to other SYSAPs anywhere in the cluster to which the SCA is connected. The CTP controller will always use the SYSAP process name "CTP\$CONTROLLER" to identify itself. The controller never accepts (or listens for) inbound connect requests.

There are three phases of controller operation, presented in the following three subsections. These are the initiating, requesting, and shutdown phases.

5.1.1 Controller To Responder Initiation

The controller establishes communications with the responder by performing an SCA connect request to the desired node, with a destination SYSAP process name of "CTP\$RESPONDER".

Flow control and credits -

A controller or a responder must allocate and extend enough credits so that all tests may be performed without hanging or failing due to a lack of resources. This is very important since CTP is NOT a symmetric protocol. To begin with, flow control is a strategy for notifying a sender of data (data being used in the broad sense of the word, not referring to block buffer transfers on the CI) how much space the receiver has so that the sender does not send more data than the receiver has buffer space for. Flow control is applied only to messages. It is the sysap protocol's responsibility to provide flow control for datagrams.

Datagrams -

If a datagram is sent to a sysap that does not have any datagram buffers contributed to its connection, the datagram may immediately be recycled to the port free datagram queue instead of being given to the process owning the connection. So it is important that enough datagrams are queued and available for a connection so that datagrams are not lost during a test case. This value should be the maximum number of datagrams plus one that may be received for any given test case. This number is seven for a VAX controller and two for a responder.

Messages -

Credits are applied directly to messages. If a credit is extended to a sysap, then it may use the credit to send a message. Since CTP is not a symmetric protocol, enough message credits should be extended to the remote sysap, to prevent a lot of scs credit messages from being sent. If only one credit was extended to a remote sysap, then an scs credit message would be sent for almost every message sent between the sysaps on that connection. The number of messages, or credits to extend should be four.

Data transfers -

To initiate a block data transfer, the local side must hold a send credit. The send credit is used for the duration of the transfer, but is available again after completion of the transfer. So for this reason, credits are also needed for buffer move operations. VAX ports allow the sysap to allocate message buffers for credits extended and use these credits for both messages and block data transfer operations. At connect time the credits extended to an HSC50 are dedicated to specific operations. HSC50 will not connect with less than 3 credits extended. The reason for this is that the first two credits are used for SNTDAT and REQDAT operations respectively. The remaining message credits are considered excess credits and are used for messages. Since CTP is based on messages it would be useless for HSC50 to accept a connect request if less than three credits have been extended (no CTP messages could be exchanged). When a host connects to an HSC50 the number of credits the host extends to HSC50 is the number of credits HSC50 will extend to the host. If an insufficient number of credits are available from HSC50's connection manager, the connect request will be rejected.

Minimum Send Credit -

This value is also called the threshold, the minimum number of message buffers the destination process should maintain for proper protocol operation. This value should be one. This tells SCS to send credit messages if the number of credits drops below the value.

The notation for offsets and constants provides both the decimal value and a symbolic name. The value is shown first, followed by a '/', followed by the symbol. The offset to the protocol type field would be 00/CTP\$CDATPTYPE, to show the symbol CTP\$CDATPTYPE with a value of zero.

00/CTP\$CDATPTYPE	Protocol type field. This is the constant 00/CTP\$CTP. (The protocol type field is intended to be compatible with all SYSAPs that exchange connect protocol data, and is subject to change until these conventions have been set up by various SYSAP developers and architects.)
01/CTP\$CDATPVERS	Protocol version number. For the protocol described in this document, the value of this constant may be determined from the cover sheet of this document. CTP\$CTPVERSION should be the value in this byte. Digital reserves all positive values, while negative values are available for customer or CSS variations of the

protocol.

02/CTP\$CDATPREV

Protocol revision number. For the protocol described in this document, the value of this constant may be determined from the cover sheet of this document. CTP\$REVISION should be the value in this byte. The value will increase by one for each revision until a new version is developed, when this value will revert to zero.

The controller, having requested a connection, should look for an accept or reject from the destination node. A rejection will cause the controller to abandon the attempted tests (with that node), reporting the failure to its caller. An acceptance will be accompanied by connect data in the same format as that sent by the controller. The protocol type and version should be examined for equality with that understood by the controller process. If the controller cannot deal with the type and version specified, then the connection should be immediately terminated through the SCA disconnect function, with a reason indicating protocol mismatch.

If the responder indicates that it uses a different version of the protocol, the responder's version will be used for the communications that will follow. It is possible for the controller to initiate a request for protocol version n, and get acceptance requiring version m. It is the responsibility of the controller to either disconnect or use the version specified by the responder.

Once the connection has been established, no special steps need be taken by the controller to maintain message or datagram buffers for incoming data. The responder will only send messages or datagrams at the request of the controller.

5.1.2 Controller To Responder Requests

Each of the functions available to the controller is provided in the form of a request message, sent to the responder, followed by some response message from the responder. While the test procedure depends upon a structure in the use of these functions, CTP does not have any functions requiring multiple messages from the controller to the responder. The following subsections deal with each function, as made available through its request message.

There are some common attributes to the request/response sequence followed by the controller. First, the request must be sent as a message except for a generate datagram request datagram. A generate datagram may be sent as either as a datagram, or as a sequenced message. The final response will come back using the initiating service, and the controller must give SCA a message or datagram buffer for the response. All other requests must be sent as sequenced messages and may be assumed to have reached their destination, or SCA will cause the connection to terminate. Datagrams may be discarded, and the controller has no way to find out if this has happened (other than no response being returned). The mandatory path is sequenced messages, unless the test being done has an explicit need for datagram communications. Datagrams and messages need only be large enough to satisfy

the needs of the CTP function contained within. Datagram and message responses should be the length of the CTP overhead plus the length of data requested.

The controller should maintain a timeout for all responses expected, since SCA communications will not always shut down spontaneously in the event of a failure. Some failures need communications to be attempted before they are detected. Thus, on timeout, the controller should attempt further communications with the node under test, forcing SCA to check the state of the system. This attempted communications can be either another request of the responder, or some other SCA action with the node under test as the destination. The timeout should be set in accordance with the system configuration. Standalone tests would have short timeouts. Since nothing should interfere with the responder, while tests done on-line would have lengthy timeouts to accomodate system activity.

All messages have a CUP\$REFERENCE field, four bytes in length. The contents of this field are unused by the responder but will be copied to all responses that are generated due to a particular request. The controller may use this to aid in correlating responses with requests, examining datagram failures, or for any other purpose desired.

All responses carry a CTP\$STATUS byte, indicating success or various types of failure. A refusal to perform the function is always indicated with the value 255/CTP\$REFUSE. The controller may always check this field on an incoming response to see if a test has been attempted that cannot be done in the current configuration or diagnosis algorithm.

Two parameters, CTP\$DELAY and CTP\$REPCOUNT, are common to several request messages. The CTP\$DELAY parameter is used to delay the action called for in the request, so the controller can set up any special conditions within the port hardware. This delay is specified in 0.5 second increments, to a maximum of 127.5 seconds. In normal use, this delay should be zero. The resolution of the clock in the timer may vary +/- one-half count, meaning that a delay request of one may be between 0.25 and 0.75 seconds in length. In addition, on-line tests may encounter random built-in queueing delays on all requests.

The CTP\$REPCOUNT parameter is used to generate multiple actions from one request. The responder will repeat the request the number of times specified in CTP\$REPCOUNT, unless an error is detected during one of the requests. In the case of the generate message and datagram requests, the responder should return the number of messages or datagrams specified by the repeat count field, AND THEN RETURN A CTP GENERATE MESSAGE or DATAGRAM RESPONSE. A count of zero in other than a message or datagram request means the responder does nothing but return a CTP response with a CTP\$NONSENSE status field value. In the case of a message or datagram request, a single datagram or message is returned as the CTP response.

The CTP\$ACTCOUNT parameter shows the actual number of times a particular request was repeated. If all is well, it will be equal to the value sent in the CTP\$REPCOUNT field. The exception is for generate message and datagram responses, in which the CTP\$ACTCOUNT should indicate starting with zero, a count of messages or datagrams already sent.

5.1.2.1 Function Set Inquiry

This function allows a controller to determine the set of functions that the responder currently implements. Responders must deal with CTP requests which are not implemented by returning the proper response with a CTP\$NOTIMP status value.

The controller simply sends a Responder Function Set Request Message, and receives a Responder Function Set Response Message with a bit mask of the functions implemented by the responder. If the bit corresponding to the CTP request opcode is set then the CTP function is implemented by the responder. If the bit is clear, then the function is not implemented. All bits relating CTP response opcodes are unused.

5.1.2.2 Buffer Map

This function allows the controller to gain access to a memory buffer in the responder system. Once the buffer has been mapped, the controller may access it via buffer move commands locally, buffer move commands to the responder, or buffer move commands to some other responder (third party I/O). The responder is only required to make a limited amount of memory resources available to any controller. If more than one controller is active on the CI at any time, then it is possible that neither controller will be able to get as much responder memory as desired. Whenever only one controller is active, then the full set of resources should be available.

Provision is made for three types of buffer allocation in the responder:

1. Data transfer verify buffers (CTP\$REALBUF), where the contents of the data will be examined to ensure proper operation.
2. Maintenance read or write verify buffers (CTP\$MAINTBUF), again with contents examined.
3. Traffic verify buffers (CTP\$FAKEBUF), where the buffer is only used to raise the level of traffic on the CI.

The responder will provide unique, real memory buffers for the first two types of requests, but may supply anything that appears to be memory in response to the third request. The reason for the third type is to allow systems to use special memory mapping techniques to minimize the actual resources used by the responder. If there is no advantage to this in a particular system, requests for CTP\$FAKEBUF will be treated the same as requests for CTP\$REALBUF.

The controller may verify the ability to move data from the responder without the ability to move data to the responder, since all real memory buffers are filled with a specified pattern upon allocation.

The controller allocates the buffers by sending a Buffer Map Request Message to the responder, and receiving a Buffer Map Response Message indicating the results of the allocation. If the buffer requested cannot be

allocated (or mapped), the CTP\$STATUS byte will be returned with a value of CTP\$NORESOURCE. Various other failures can also occur. Otherwise, the name of the buffer, and actual size allocated, will be returned. For CTP\$REALBUF and CTP\$FAKEBUF requests, the name will be in the format expected by the SCA read and write functions, while the name returned from CTP\$MAINTBUF requests will be in the format expected by the responder's port when maintenance read or write requests arrive.

All mapped buffers will remain valid until a controller requests the buffers be unmapped, or until the connection over which the buffers were mapped is terminated or broken.

5.1.2.3 Buffer Unmap

This function allows the controller to release the resources allocated by the buffer map function. Release of part of a buffer is not allowed. The buffer name and size are placed in a Buffer Unmap Request Message, along with the buffer type, and sent to the responder. A Buffer Unmap Response Message is returned, with CTP\$BUFTYPE set to CTP\$NOBUFFER if the buffer type, name, and size do not match any currently allocated buffer. Once the response is received, the controller can presume the resources are available for other uses.

5.1.2.4 Move Buffer

This function allows the controller to direct data movement to or from the responder, with the responder's port initiating the block transfer. A Move Buffer Request Message is sent to the responder with the parameters of the move. The controller indicates the transfer function in the CTP\$MOVETYPE field. The responder node is always at one end of the transfer, while the other end is specified in CTP\$OTHERNODE as a CI port address. The buffer name, offset, and size parameters in the message will be used as sent when the responder calls the appropriate SCA read or write function.

The responder will return a Move Buffer Response Message when either all repetitions of the move have completed, or a transfer ends in error.

For third party buffer transfers it is necessary for connections to be established between responders. This is accomplished by using the CTP\$CONNECTREQ request message. For third party connections, responder to responder, the initiating responder should extend three credits to the responder it is connecting to. There is no third party disconnect request. All third party connections are disconnected when there is no connection to any controller.

5.1.2.5 Generate Message

This function causes the responder to create a stream of sequenced messages directed to the controller. These messages will be sent over the connection that is being used for requests and responses. Generate Message requests must be made using messages. The controller sends a Generate Message Request Message, with the size and data pattern information reflecting the message desired. Each response message is a carbon copy of the final response message, with the exception of the actual count field. This field starts with zero and is incremented for each subsequent message until all messages requested through the repeat count have been returned. The length of a message should be the CTP overhead plus the length of data requested.

The number of Generate Message Response Messages received by the controller will depend upon the repeat count specified. The actual number of messages returned by the responder will be one more than the repeat count, because the final message (the CTP response) is not considered as part of the message stream. Each response will have the number of responses generated so far in the CTP\$ACTCOUNT field. I.e., the first will be 0, the second 1, and so on until the last.

5.1.2.6 Generate Datagram

This function causes the responder to create a stream of datagrams directed to the controller. These datagrams will be sent over the connection that is being used for requests and responses. If the request comes in as a sequenced message, the number of datagrams returned in the stream will be equal to the repeat count, and the final response will be sent out as a message. If the request comes in as a datagram, the number of datagrams returned in the stream will be equal to the repeat count, plus the final response which will be sent out as a datagram. In all cases, a message request results in a final message response and a datagram request results in a final datagram response. The Generate datagram request is the only action that may be requested using a datagram. The controller sends a Generate Datagram Request Datagram or Message, with the size and data pattern information reflecting the datagram desired. Each response datagram is a carbon copy of the final response datagram, with the exception of the actual count field. This field starts with zero and is incremented for each subsequent datagram until all datagram requested through the repeat count have been returned. The length of a datagram should be the CTP overhead plus the length of data requested.

Each response will have the number of responses generated so far in the CTP\$ACTCOUNT field. I.e., the first will be 0, the second 1, and so on until the final response which will be the repeat count.

5.1.2.7 Generate Reset

NOTE: This function may be destructive to the target node, and should be used with caution.

This function will cause the responder to generate a specific reset packet on the CI. The controller sends a Generate Reset Request Message, with the CTP\$OTHERNODE field set to the CI port address of the target node. The CTP\$EXTEND field is set to the type of reset to be done, conditional or forced.

When the responder has completed sending the reset packet on the CI, it will return a Generate Reset Response Message.

5.1.2.8 Generate Start

This function will cause the responder to generate a specific start packet on the CI. The controller sends a Generate Start Request Message, with the CTP\$OTHERNODE field set to the CI port address of the target node. The CTP\$EXTEND field is set to indicate if the start is at the node's default or the packet's specified address. The CTP\$STARTADR field contains the node-specific start address.

When the responder has completed sending the start packet on the CI, it will return a Generate Start Response Message.

5.1.2.9 Configuration Data

This function allows the controller to examine the information used by the responder's SCA modules. The controller generates a Configuration Data Request Message, with the CTP\$OTHERNODE field set to indicate the pairing for the configuration data.

The responder will get the data from its SCA, and return a Configuration Data Response Message.

5.1.2.10 CI Port Counter Read

This function allows the controller to examine the counters available in the responder's CI port. The controller sends a Counter Read Request Message, indicating the number of the port to be used for future counts.

The responder will read and reset the counters, and return a Counter Read Response Message.

This request must be used in pairs of two. Some nodes may read the specified nodes counters immediately, while others may read the preset values from the last read counter request. To avoid problems with these two types of implementations, the counters should be first read with the desired node number to reset the counters and/or set the counters to count the

appropriate ports transactions. Then at a later time the counters should be reread using the desired node number again to assure proper results. This process should be repeated for all subsequent counter read requests.

5.1.3 Controller To Responder Shutdown

The controller has two possible intentions in a shutdown of communications with the responder. First, an immediate shutdown may be desired, where the state and orderly completion of outstanding requests is not important. In this case, the controller may simply call the SCA disconnect service, providing the appropriate reason code.

Second, an orderly shutdown may be desired. In this case, the controller will send a Listener Disconnect Request Message. The controller then receives the Listener Disconnect Response Message from the responder. At this point, the controller calls the SCA disconnect service, and the link is terminated.

It is possible for the responder to shut down communications. This is not a normal situation, and usually reflects an error condition within the SCA. The controller must be able to deal with this situation, re-establishing communications with the responder if needed. Also, the controller will lose any buffers that have been allocated, with the possible exception of maintenance buffers. See the section on Buffer Map operation for details.

5.2 CTP Responder Operation

The responder may at times, have many requests to service simultaneously. The following describes the preferred method for determining the order in which to respond to a group of requests. The method described is not the only way a responder may operate on requests. The intention is to achieve responder operation such that one controller does not impede the operation of another controller connected to the same responder.

The method is a round robin scheme on a connection basis. All requests received on different connections should have an equal opportunity to be performed, without having to wait for multiple requests from a single connection to be fully performed.

There are two cases, first two or more (if message credits exist) request messages could be received by a responder on one connection (this connection may be to either another responder or a controller). Assuming that a third request is received from another connection before the first two commands have been executed, the command from the second connection should not have to wait for execution of the first two commands before being serviced. i.e. the second connection's request should be treated with equal priority as the first connection's request. So the first connection's first command should be executed and then the second connection's command followed by the first connection's second command.

The second case is similar to the first except that the first connection has a command with a repeat count greater than one. This in effect is the same as the first connection having more than one command outstanding and should be treated as described above. The first connections first repetition of the command should be executed, followed by the second connection's command, and then subsequent iterations of the first command.

The basic rules are that commands received on a connection are executed sequentially, while commands received on different connection are executed in a round robin fashion. This method need only be employed to the extent of obtaining an even distribution of responder responses to all controllers connected to the responder. It is possible that a responder will not return requests in the order requested if more than one request is sent to the responder. The following are some examples that should help clarify the method.

Example One:

The responder receives two requests, one right after the other, each on a different connection. The first command will be labeled command A, the second command B.

Command A - CTP Generate message request message, repeat count = 3.
Command B - CTP Function set request message.

The responder should execute these requests in the following order:

Command A, first iteration.
Command B.
Command A, second iteration.
Command A, third iteration.
Command A, response message.

Example Two:

The responder receives three requests, one right after the other, two on one connection and one on another connection. The commands are as follows:

Command A - CTP Function set request message.
Command B - CTP Generate message request message.
Command C - CTP Generate message request message.

Command A and B are from one connection and command C is from the other connection. The responder should execute these requests in the following order:

Command A.
Command C.
Command B.

Example Three:

This example is a more complicated example and is intended to show worst case. The responder receives three requests, one right after the other, two on one connection and one on another connection. At a later time while the first three commands are executing, a fourth command is received and is to be executed. The first three commands are as follows:

Command A - CTP Generate message request message with a repeat count of 3.
Command B - CTP function set request message.
Command C - CTP Generate message request message with a repeat count of 5.

Command A and B are from the same connection and command C is from another connection. The responder should begin execution in the following order:

Command A, first iteration.
Command C, first iteration.
Command A, second iteration.
Command C, second iteration.

At this point, a fourth command is received, Command D. This command is from a third connection. The command is to do a data transfer. Execution should proceed as follows:

Command A, third iteration.
Command C, third iteration.
Command D.
Command A, final response.
Command C, fourth iteration.
Command B.
Command C, fifth iteration.
Command C, final response.

6.0 REQUIRED/OPTIONAL CTP REQUESTS

In order to participate in a minimal set of CI cluster tests, all responders are required to implement certain CTP functions. Additional CTP functions must also be implemented depending upon the functionality of the responder's port. This section lists the base set of functions that MUST be implemented by all responders, and the additional functions that may also have to be implemented.

Upon receiving a CTP request that is not implemented, the responder should return the appropriate CTP response with a CTP\$NOTIMP status value.

The following CTP packet types must be implemented by all responders:

- CTP FUNCTION SET
- CTP GENERATE MESSAGE
- CTP GENERATE DATAGRAM
- CTP READ COUNTERS
- CTP CONFIGURATION DATA
- CTP LISTENER DISCONNECT
- CTP STOP/START UNSOLICITED ACTIVITY (to be used by CINT only)

Responders must implement an additional CTP function if the corresponding CI function is implemented by the responder's port. The following chart shows the additional responder functions that must be implemented as a function of the options implemented by the CI port.

CI PORT OPTION	CTP FUNCTION
PACKETS SND/RCV	
SNTDAT/CNF	CTP\$BUFMAP, CTP\$BUFUNM, CTP\$MOVBUF
CNF/SNTDAT	CTP\$BUFMAP, CTP\$BUFUNM
DATREQ/RETDAT	CTP\$BUFMAP, CTP\$BUFUNM, CTP\$MOVBUF
RETDAT/DATREQ	CTP\$BUFMAP, CTP\$BUFUNM
IDREQ/ID	
SNTMDAT/MCNF	CTP\$MBUFMAP, CTP\$MBUFUNM
MCNF/SNTMDAT	CTP\$MBUFMAP, CTP\$MBUFUNM, CTP\$MOVMBUF, CTP\$MSTATE
MDATREQ/RETMDAT	CTP\$MBUFMAP, CTP\$MBUFUNM, CTP\$MOVMBUF, CTP\$MSTATE
RETMDAT/MDATREQ	CTP\$MBUFMAP, CTP\$MBUFUNM
RST/	CTP\$GENRST

CI CLUSTER TEST PROTOCOL
REQUIRED/OPTIONAL CTP REQUESTS

Page 23
12 Apr 83

SEQ 0055

STRT/	CTP\$GENSTR
/RST-STRT	CTP\$MSTATE
LP/LP	
STATES IMPLEMENTED	
UNINITIALIZED	
UNINIT/MAINT	
DISABLED	
DSBLD/MAINT	
ENABLED	
ENABLED/MAINT	

7.0 CTP MESSAGE FORMATS

This section describes the formats of the messages that are passed between processes speaking CTP. All of the messages are sent using the SCA, and this section does not address the SCA portions of the messages. Refer to the SCA documents for those formats.

The preceding section described the use of the messages shown in this section. This section merely presents the encoding formats and syntax of the messages.

Note that all messages begin with a one-byte opcode, indicating what type the message is. Digital reserves opcodes in the range 0 to 127, while values in the range 128 to 254 are reserved for customer and CSS additions to the protocol. Opcodes generated by the responder as a result of a message from the controller will be 64 greater than the controller's opcode. The opcode value 255 is reserved for extending the opcode space to multiple byte values.

All messages have a four-byte reference number in the second through fifth bytes. This number is not used by the responder and will be copied from a request message to all response messages generated by that request.

All responses contain a status code in the sixth byte. Various possibilities for success, failure and refusal can be described by the responder in the status field. In all cases, successful status values are indicated using positive byte values in the status field, while negative status field values indicate a failure or refusal on the part of the responder.

7.1 Responder Function Set Request Message

00/CTP\$OPCODE	1 Byte, value is 00/CTP\$FUNCTREQ.
01/CTP\$REFERENCE	4 Bytes, contains reference number.

7.2 Responder Function Set Response Message

00/CTP\$OPCODE	1 Byte, value is 64/CTP\$FUNCTRSP.
01/CTP\$REFERENCE	4 Bytes, contains reference number.
05/CTP\$STATUS	1 Byte, value is 00/CTP\$SUCCESS.
06/CTP\$FMASK	32 Bytes, arranged as a 256 bit array, with a bit set corresponding to the CTP request opcode if the CTP function is implemented by the responder. If the bit is clear, then the function is not implemented. All bits relating CTP response opcodes are unused. The least significant bit of the first byte corresponds to the function with request opcode zero, and the

most significant bit of the 32nd byte corresponds to the function with request opcode 255. For CTP functions not implemented, the responder must still be able to receive the CTP packet type, but should return the response with a NOT IMPLEMENTED status value.

7.3 Buffer Map Request Message

00/CTP\$OPCODE 1 Byte, value is 01/CTP\$BUFMAPREQ.

01/CTP\$REFERENCE 4 Bytes, contains reference number.

05/CTP\$BUFTYPE 1 Byte indicating type of buffer to
 allocate, values are:

 01/CTP\$REALBUF Real Memory Buffer.
 02/CTP\$FAKEBUF 'Black Hole' Buffer.

06/CTP\$RESERV06 2 Bytes, must be zero.

08/CTP\$GENFUNCT 1 Byte specifying the function to use for the
 initial contents of the allocated buffer.
 Values are:

 00/CTP\$GENFFILL Fill buffer with constant.
 01/CTP\$GENFBPAIR Fill w/byte pair numbers.
 02/CTP\$GENFNODE Fill w/responder node number

09/CTP\$GENCONST 1 Byte holding the fill constant, only used
 with the 00/CTP\$GENFFILL function. May be any
 byte value.

10/CTP\$PKTSIZ 1 Byte, contains the base packet size to use
 for the transfer.

 00/CTP\$PS512 Size is a multiple of 512 bytes.
 01/CTP\$PS576 Size is a multiple of 576 bytes.

11/CTP\$PKTMULT 1 Byte, specifying the packet size
 multiple.

12/CTP\$BUFLNGTH 4 Byte integer indicating number of bytes to
 allocate.

7.4 Buffer Map Response Message

00/CTP\$OPCODE 1 Byte, value is 65/CTP\$BUFMAPRSP.

01/CTP\$REFERENCE 4 Bytes, contains reference number.

05/CTP\$STATUS 1 Byte indicating type of buffer allocated,
 values are:

 01/CTP\$REALBUF Real Memory Buffer.

 02/CTP\$FAKEBUF "Black Hole" Buffer.

 250/CTP\$BUFLIMIT Buffer Map Limit Exceeded.

 253/CTP\$NORESOURCE Resource Failure.

 254/CTP\$NONSENSE Request Not Understood.

 255/CTP\$NOTIMP Function Not Implemented.

06/CTP\$RESERV06 2 Bytes, must be zero.

08/CTP\$GENFUNCT 1 Byte specifying the function to use for the
 initial contents of the allocated buffer.
 Values are:

 00/CTP\$GENFFILL Fill buffer with constant.

 01/CTP\$GENFBPAIR Fill w/byte pair numbers.

 02/CTP\$GENFNODE Fill w/responder node number

09/CTP\$GENCONST 1 Byte holding the fill constant, only used
 with the 00/CTP\$GENFFILL function. Must be byte
 value supplied in request.

10/CTP\$PKTSIZ 1 Byte, contains the base packet size to used
 for the transfer.

 00/CTP\$PS512 Size is a multiple of 512 bytes.

 01/CTP\$PS576 Size is a multiple of 576 bytes.

11/CTP\$PKTMULT 1 Byte, specifying the packet size
 multiple. Must be value supplied in request.

12/CTP\$BUFLNGTH 4 Byte integer with number of bytes actually
 allocated.

16/CTP\$BUFLNAME 4 Byte value with the name of the buffer
 allocated.

7.5 Buffer Unmap Request Message

00/CTP\$OPCODE 1 Byte, value is 02/CTP\$BUFUNMREQ.

01/CTP\$REFERENCE 4 Bytes, contains reference number.

05/CTP\$BUFTYPE 1 Byte indicating type of buffer to
 deallocate, must be the type allocated, values:

 01/CTP\$REALBUF Real Memory Buffer.
 02/CTP\$FAKEBUF 'Black Hole' Buffer.

06/CTP\$RESERV6 3 Bytes, must be zero.

09/CTP\$RESERV9 1 Byte, must be zero.

10/CTP\$RESERV10 2 Bytes, must be zero.

12/CTP\$BUFLNGTH 4 Byte integer indicating number of bytes to
 release. Must be equal to the size of the
 buffer originally mapped.

16/CTP\$BUFLNAME 4 Byte value with the name of the buffer to
 release. Must be equal to the name of the
 buffer mapped.

7.6 Buffer Unmap Response Message

00/CTP\$OPCODE 1 Byte, value is 66/CTP\$BUFUNMRSP.

01/CTP\$REFERENCE 4 Bytes, contains reference number.

05/CTP\$STATUS 1 Byte indicating type of buffer deallocated,
 values are:

 01/CTP\$REALBUF Real Memory Buffer.
 02/CTP\$FAKEBUF 'Black Hole' Buffer.
 249/CTP\$NOBUFMAPPED No Buffer Mapped.
 253/CTP\$NORESOURCE Resource Failure.
 254/CTP\$NONSENSE Request Not Understood.
 255/CTP\$NOTIMP Function Not Implemented.

06/CTP\$RESERV6 3 Bytes, must be zero.

09/CTP\$RESERV9 1 Byte, must be zero.

10/CTP\$RESERV10 2 Bytes, must be zero.

12/CTP\$BUFLNGTH 4 Byte integer indicating number of bytes
 released.

16/CTP\$BUFLNAME 4 Byte value with the name of the buffer
 released.

7.7 Move Buffer Request Message

00/CTP\$OPCODE 1 Byte, value is 03/CTP\$MOVBUFREQ.
01/CTP\$REFERENCE 4 Bytes, contains reference number.
05/CTP\$DELAY 1 Byte specifying delay prior to action
 (~500ms/count).
06/CTP\$REPCOUNT 2 Byte integer containing a repeat count.
08/CTP\$MOVETYPE 1 Byte specifying the type of buffer move to
 perform:
 00/CTP\$MOVEFROM Responder does a read.
 01/CTP\$MOVETO Responder does a write.
09/CTP\$OTHERNODE 1 Byte containing the number of the other node
 involved.
10/CTP\$PKTSIZ 1 Byte, contains the base packet size to use
 for the transfer.
 00/CTP\$PS512 Size is a multiple of 512 bytes.
 01/CTP\$PS576 Size is a multiple of 576 bytes.

11/CTP\$PKTMULT 1 Byte, specifying the packet size multiple.
12/CTP\$BUFLENGTH 4 Byte integer containing the size of the move
 in bytes.
16/CTP\$BUFLNAME 4 Byte value with the local buffer name.
20/CTP\$BUFLOFSET 4 Byte value containing the local buffer
 offset.
24/CTP\$BUFRNAME 4 Byte value with the remote buffer name.
28/CTP\$BUFROFSET 4 Byte value containing the remote buffer
 offset.

7.8 Move Buffer Response Message

00/CTP\$OPCODE 1 Byte, value is 67/CTP\$MOVBUFRSP.
01/CTP\$REFERENCE 4 Bytes, contains reference number.
05/CTP\$STATUS 1 Byte with the SCA or CTP status of the last
 move performed. Values are:
 00/CTP\$SUCCESS Function Successful.

246/CTP\$NOCONECT No connection exists.
247/CTP\$PKTSIZFAIL Unable to use specified packet size.
249/CTP\$NOBUFMAPPED No Buffer Mapped.
251/CTP\$FAILURE Function Failure.
252/CTP\$REFUSE Request refused.
253/CTP\$NORESOURCE Resource Failure.
254/CTP\$NONSENSE Request Not Understood.
255/CTP\$NOTIMP Function Not Implemented.

06/CTP\$ACTCOUNT 2 Byte integer holding the actual number of moves done.

08/CTP\$MOVETYPE 1 Byte specifying the type of buffer move performed:

00/CTP\$MOVEFROM Responder did a read.
01/CTP\$MOVETO Responder did a write.

09/CTP\$OTHERNODE 1 Byte containing the number of the other node involved.

10/CTP\$PKTSIZ 1 Byte, contains the base packet size to used for the transfer. Same as specified in request.

00/CTP\$PS512 Size is a multiple of 512 bytes.
01/CTP\$PS576 Size is a multiple of 576 bytes.

11/CTP\$PKTMULT 1 Byte, specifying the packet size multiple. Same as specified in request.

12/CTP\$BUFLENGTH 4 Byte integer containing the size of the move in bytes. Same as specified in request.

16/CTP\$BUFLNAME 4 Byte value with the local buffer name.

20/CTP\$BUFLOFSET 4 Byte value containing the local buffer offset.

24/CTP\$BUFRNAME 4 Byte value with the remote buffer name.

28/CTP\$BUFROFSET 4 Byte value containing the remote buffer offset.

7.9 Generate Message Request Message

00/CTP\$OPCODE 1 Byte, value is 04/CTP\$GENMSGREQ.

01/CTP\$REFERENCE 4 Bytes, contains reference number.

05/CTP\$DELAY 1 Byte specifying delay prior to action (~500ms/count).

06/CTP\$REPCOUNT 2 Byte integer containing a repeat count.

08/CTP\$GENFUNCT 1 Byte specifying the function to use for the
data field in the generated packets. Values
are:

 00/CTP\$GENFFILL Fill packet with constant.

 01/CTP\$GENFBPAIR Fill w/byte pair numbers.

 02/CTP\$GENFNODE Fill w/responder node number

 03/CTP\$GENFIMAGE Fill w/data in this packet.

09/CTP\$GENCONST 1 Byte holding a fill constant if
CTP\$GENFFILL, or zero otherwise. May be any
byte value.

10/CTP\$RESERV10 2 Bytes must be zero.

12/CTP\$GENLENGTH 2 Byte integer indicating the generated
message length.

14/CTP\$IMAGEDATA N bytes of image function data if appropriate.

7.10 Generate Message Response Message

00/CTP\$OPCODE 1 Byte, value is 68/CTP\$GENMSGGRSP.

01/CTP\$REFERENCE 4 Bytes, contains reference number.

05/CTP\$STATUS 1 Byte with the SCA or CTP status of the last
message sent. Values are:

 00/CTP\$SUCCESS Function Successful.

 253/CTP\$NORESOURCE Resource Failure.

 254/CTP\$NONSENSE Request Not Understood.

06/CTP\$ACTCOUNT 2 Byte integer containing the count of
messages sent.

08/CTP\$GENFUNCT 1 Byte specifying the function used for the
data field in the generated packets. Values
are:

 00/CTP\$GENFFILL Fill packet with constant.

 01/CTP\$GENFBPAIR Fill w/byte pair numbers.

 02/CTP\$GENFNODE Fill w/responder node number

 03/CTP\$GENFIMAGE Fill w/data in this packet.

09/CTP\$GENCONST 1 Byte holding a fill constant if
CTP\$GENFFILL, or zero otherwise. Must be byte
value supplied in request.

10/CTP\$RESERV10 2 Bytes must be zero.

12/CTP\$GENLENGTH 2 Byte integer indicating the generated message length.

14/CTP\$GENDATA N Bytes containing the generated data.

7.11 Generate Datagram Request Message

00/CTP\$OPCODE 1 Byte, value is 05/CTP\$GENDGRREQ.

01/CTP\$REFERENCE 4 Bytes, contains reference number.

05/CTP\$DELAY 1 Byte specifying delay prior to action (~500ms/count).

06/CTP\$REPCOUNT 2 Byte integer containing a repeat count.

08/CTP\$GENFUNCT 1 Byte specifying the function to use for the data field in the generated packets. Values are:

00/CTP\$GENFFILL	Fill packet with constant.
01/CTP\$GENFBPAIR	Fill w/byte pair numbers.
02/CTP\$GENFNODE	Fill w/responder node number
03/CTP\$GENFIMAGE	Fill w/data in this packet.

09/CTP\$GENCONST 1 Byte holding a fill constant, if appropriate. May be any byte value.

10/CTP\$RESERV10 2 Bytes must be zero.

12/CTP\$GENLENGTH 2 Byte integer indicating the generated datagram length.

14/CTP\$IMAGEDATA N bytes of image function data if appropriate.

7.12 Generate Datagram Response Message

00/CTP\$OPCODE 1 Byte, value is 69/CTP\$GENDGRRSP.

01/CTP\$REFERENCE 4 Bytes, contains reference number.

05/CTP\$STATUS 1 Byte with the SCA or CTP status of the last datagram sent. Values are:

00/CTP\$SUCCESS	Function Successful.
253/CTP\$NORESOURCE	Resource Failure.
254/CTP\$NONSENSE	Request Not Understood.

06/CTP\$ACTCOUNT 2 Byte integer containing the count of datagrams sent.

08/CTP\$GENFUNCT 1 Byte specifying the function to use for the

data field in the generated packets. Values are:

00/CTP\$GENFFILL	Fill packet with constant.
01/CTP\$GENFBPAIR	Fill w/byte pair numbers.
02/CTP\$GENFNODE	Fill w/responder node number
03/CTP\$GENFIMAGE	Fill w/data in this packet.

09/CTP\$GENCONST	1 Byte holding a fill constant if CTP\$GENFFILL, or zero otherwise. Must be byte value supplied in request.
10/CTP\$RESERV10	2 Bytes must be zero.
12/CTP\$GENLENGTH	2 Byte integer indicating the generated datagram length.
14/CTP\$GENDATA	N Bytes containing the generated data.

7.13 Generate Reset Request Message

C0/CTP\$OPCODE	1 Byte, value is 06/CTP\$GENRSTREQ.
01/CTP\$REFERENCE	4 Bytes, contains reference number.
05/CTP\$DELAY	1 Byte specifying delay prior to action (~500ms/count).
06/CTP\$REPCOUNT	2 Byte integer containing a repeat count.
08/CTP\$EXTEND	1 Byte specifying whether the reset is conditional:
00/CTP\$UNCONDRST	Forced reset.
01/CTP\$CONDRST	Conditionally reset.

09/CTP\$OTHERNODE	1 Byte containing the number of the node to reset.
-------------------	--

7.14 Generate Reset Response Message

00/CTP\$OPCODE	1 Byte, value is 70/CTP\$GENRSTRSP.
01/CTP\$REFERENCE	4 Bytes, contains reference number.
05/CTP\$STATUS	1 Byte with the SCA or CTP status of the last reset request. Values are:

00/CTP\$SUCCESS	Function Successful.
251/CTP\$FAILURE	Function Failure.
252/CTP\$REFUSE	Request refused.
253/CTP\$NORESOURCE	Resource Failure.
254/CTP\$NONSENSE	Request Not Understood.
255/CTP\$NOTIMP	Function Not Implemented.

06/CTP\$ACTCOUNT 2 Byte integer containing number of resets sent.

08/CTP\$EXTEND 1 Byte specifying whether the reset was conditional:

00/CTP\$UNCONDRST	Forced reset.
01/CTP\$CONDRST	Conditionally reset.

09/CTP\$OTHERNODE 1 Byte containing the number of the node that was reset.

7.15 Generate Start Request Message

00/CTP\$OPCODE 1 Byte, value is 07/CTP\$GENSTRREQ.

01/CTP\$REFERENCE 4 Bytes, contains reference number.

05/CTP\$DELAY 1 Byte specifying delay prior to action (~500ms/count).

06/CTP\$REPCOUNT 2 Byte integer containing a repeat count.

08/CTP\$EXTEND 1 Byte specifying whether the start address is default:

00/CTP\$DEFAULTADR	Use node's default address.
01/CTP\$SPECIFADR	Use the specified address.

09/CTP\$OTHERNODE 1 Byte containing the number of the node to start.

10/CTP\$RESERV10 2 Bytes, must be zero.

12/CTP\$STARTADR 4 Byte value containing a node-specific start address.

7.16 Generate Start Response Message

00/CTP\$OPCODE 1 Byte, value is 71/CTP\$GENSTRRSP.

01/CTP\$REFERENCE 4 Bytes, contains reference number.

05/CTP\$STATUS 1 Byte with the SCA or CTP status of the last
start sent. Values are:

00/CTP\$SUCCESS	Function Successful.
251/CTP\$FAILURE	Function Failure.
252/CTP\$REFUSE	Request refused.
253/CTP\$NORESOURCE	Resource Failure.
254/CTP\$NONSENSE	Request Not Understood.
255/CTP\$NOTIMP	Function Not Implemented.

06/CTP\$ACTCOUNT 2 Byte integer containing number of starts
sent.

08/CTP\$EXTEND 1 Byte specifying whether start address was
defaulted:

00/CTP\$DEFAULTADR	Node's default was used.
01/CTP\$SPECIFADR	Specified address was used.

09/CTP\$OTHERNODE 1 Byte containing the number of the node
started.

10/CTP\$RESERV10 2 Bytes, must be zero.

12/CTP\$STARTADR 4 Byte value containing the node-specific
start address.

7.17 Start / Stop Unsolicited Activity Request Message

00/CTP\$OPCODE 1 Byte, value is 08/CTP\$NOACTREQ.

01/CTP\$REFERENCE 4 Bytes, contains reference number.

05/CTP\$RESERV5 4 Bytes, must be zero.

09/CTP\$RESERV9 1 Byte, must be zero.

10/CTP\$NOACTFLAG 1 Byte contains flag to either stop or start
normal unsolicited activity.

00/CTP\$NOACTON	Turn normal unsolicited activity on.
01/CTP\$NOACTOFF	Turn normal unsolicited activity off.

WARNING: If the poller has been stopped and the
connection to the controller that requested the
stop is lost, it is the responsibility of the
responder to automatically restart the poller.

7.18 Start / Stop Unsolicited Activity Response Message

00/CTP\$OPCODE 1 Byte, value is 72/CTP\$NOACTRSP.
01/CTP\$REFERENCE 4 Bytes, contains reference number.
05/CTP\$STATUS 1 Byte indicating status of request:
 00/CTP\$SUCCESS Function Successful.
 251/CTP\$FAILURE Function Failure.
 252/CTP\$REFUSE Request refused.
 254/CTP\$NONSENSE Request Not Understood.
 255/CTP\$NOTIMP Function Not Implemented.

7.19 Configuration Data Request Message

00/CTP\$OPCODE 1 Byte, value is 09/CTP\$CONFIGREQ.
01/CTP\$REFERENCE 4 Bytes, contains reference number.
05/CTP\$RESERV5 4 Bytes, must be zero.
09/CTP\$OTHERNODE 1 Byte containing number of node for which
 configuration information is desired.

7.20 Configuration Data Response Message

00/CTP\$OPCODE 1 Byte, value is 73/CTP\$CONFIGRSP.
01/CTP\$REFERENCE 4 Bytes, contains reference number.
05/CTP\$STATUS 1 Byte indicating knowledge of specified node:
 00/CTP\$SUCCESS Function Successful.
 248/CTP\$NODEUNKNOWN Node Unknown To SCA.
 253/CTP\$NORESOURCE Resource Failure.
 254/CTP\$NONSENSE Request Not Understood.

06/CTP\$RESERV6 3 Bytes, must be zero.
09/CTP\$OTHERNODE 1 Byte containing number of node
11/CTP\$CFGPOSTS 1 Byte containing path 0 status.
 00/CTP\$PATHBAD Path is marked bad.
 01/CTP\$PATHGOOD Path is marked good.

12/CTP\$CFGPOSTS 1 Byte containing path 1 status.

00/CTP\$PATHBAD Path is marked bad.
01/CTP\$PATHGOOD Path is marked good.

7.21 Counter Read Request Message

00/CTP\$OPCODE 1 Byte, value is 10/CTP\$COUNTSREQ.
01/CTP\$REFERENCE 4 Bytes, contains reference number.
05/CTP\$RESERV5 4 Bytes, must be zero.
09/CTP\$OTHERNODE 1 Byte containing number of node which future counts should monitor or which counts are read.
255/CTP\$ALLNODES Count for all nodes.
10/CTP\$CNTFLG 1 Byte indicating to keep or release control of counters. Values are:
00/CTP\$RELCNT Release counters.
01/CTP\$KEEPCNT Keep counters.

7.22 Counter Read Response Message

00/CTP\$OPCODE 1 Byte, value is 74/CTP\$COUNTSRSP.
01/CTP\$REFERENCE 4 Bytes, contains reference number.
05/CTP\$STATUS 1 Byte with the CTP status. Values are:
00/CTP\$SUCCESS Function Successful.
253/CTP\$NORESOURCE Resource Failure.
254/CTP\$NONSENSE Request Not Understood.
06/CTP\$RESERV6 3 Bytes, must be zero.
09/CTP\$OTHERNODE 1 Byte containing number of node which future counts will monitor.
10/CTP\$RESERV10 2 Bytes, must be zero.
12/CTP\$CNTRPOACK 4 Bytes containing total ACKs received on path 0.
16/CTP\$CNTRPONAK 4 Bytes containing total NAKs received on path 0.
20/CTP\$CNTRPONRSP 4 Bytes containing total no responses received on path 0.

24/CTP\$CNTRP1ACK	4 Bytes containing total ACKs received on path 1.
28/CTP\$CNTRP1NAK	4 Bytes containing total NAKs received on path 1.
32/CTP\$CNTRP1NRSP	4 Bytes containing total no responses received on path 1.
36/CTP\$CNTRDISCDG	4 Bytes containing total number of Discarded datagrams.

7.23 Connect Request Message

00/CTP\$OPCODE	1 Byte, value is 11/CTP\$CONNECTREQ.
01/CTP\$REFERENCE	4 Bytes, contains reference number.
05/CTP\$RESERV5	4 Bytes, must be zero.
09/CTP\$OTHERNODE	1 Byte containing the node to connect to.

Note: For third party connections, responder to responder, the initiating responder should allocate two message buffers locally, such that the remote responder is extended these credits. The initiating responder should also specify a credit threshold such that the remote responder extends a minimum of two message credits to the initiating responder.

7.24 Connect Response Message

00/CTP\$OPCODE	1 Byte, value is 75/CTP\$CONNECTRSP.
01/CTP\$REFERENCE	4 Bytes, contains reference number.
05/CTP\$STATUS	1 Byte, contains status of requested operation.

00/CTP\$SUCCESS	Function Successful.
245/CTP\$CONTRANS	Connection is in state of transition.
248/CTP\$NODEUNKNOWN	Node Unknown To SCA.
251/CTP\$FAILURE	Function Failure.
253/CTP\$NORESOURCE	Resource Failure.
254/CTP\$NONSENSE	Request Not Understood.

7.25 Listener Disconnect Request Message

00/CTP\$OPCODE 1 Byte, value is 12/CTP\$FINISHREQ.
01/CTP\$REFERENCE 4 Bytes, contains reference number.

7.26 Listener Disconnect Response Message

00/CTP\$OPCODE 1 Byte, value is 76/CTP\$FINISHRSP.
01/CTP\$REFERENCE 4 Bytes, contains reference number.
05/CTP\$STATUS 1 Byte, value must be 00/CTP\$SUCCESS.

7.27 Maintenance Buffer Map Request Message

00/CTP\$OPCODE 1 Byte, value is 13/CTP\$MBUFMAPREQ.
01/CTP\$REFERENCE 4 Bytes, contains reference number.
05/CTP\$BUFTYPE 1 Byte indicating type of buffer to allocate.
 03/CTP\$MAINTBUF Maintenance Data Buffer.

06/CTP\$RESERV06 2 Bytes, must be zero.
08/CTP\$GENFUNCT 1 Byte specifying the function to use for the
 initial contents of the allocated buffer.
 Values are:
 00/CTP\$GENFFILL Fill buffer with constant.
 01/CTP\$GENFBPAIR Fill w/byte pair numbers.
 02/CTP\$GENFNODE Fill w/responder node number

09/CTP\$GENCONST 1 Byte holding the fill constant, only used
 with the 00/CTP\$GENFFILL function. May be any
 byte value.

10/CTP\$PKTSIZ 1 Byte, contains the packet size to use for
 the transfer.
 00/CTP\$PS512 Size is 512 bytes.
 01/CTP\$PS576 Size is 576 bytes.

11/CTP\$RESERV11 1 Byte, must be zero.
12/CTP\$BUFLNGTH 4 Byte integer indicating number of bytes to
 allocate.

7.28 Maintenance Buffer Map Response Message

00/CTP\$OPCODE 1 Byte, value is 77/CTP\$MBUFMAPRSP.
01/CTP\$REFERENCE 4 Bytes, contains reference number.
05/CTP\$STATUS 1 Byte indicating type of buffer allocated,
 values are:

00/CTP\$SUCCESS Function Successful.
250/CTP\$BUFLIMIT Buffer Map Limit Exceeded.
253/CTP\$NORESOURCE Resource Failure.
254/CTP\$NONSENSE Request Not Understood.
255/CTP\$NOTIMP Function Not Implemented.

06/CTP\$RESERV06 2 Bytes, must be zero.

08/CTP\$GENFUNCT 1 Byte specifying the function to use for the
 initial contents of the allocated buffer.
 Values are:

00/CTP\$GENFFILL Fill buffer with constant.
01/CTP\$GENFBPAIR Fill w/byte pair numbers.
02/CTP\$GENFNODE Fill w/responder node number

09/CTP\$GENCONST 1 Byte holding the fill constant, only used
 with the 00/CTP\$GENFFILL function. Must be byte
 value supplied in request.

10/CTP\$PKTSIZ 1 Byte, contains the base packet size to used
 for the transfer.

00/CTP\$PS512 Size is 512 bytes.
01/CTP\$PS576 Size is 576 bytes.

11/CTP\$RESERV11 1 Byte, must be zero.

12/CTP\$BUFLNGTH 4 Byte integer with number of bytes actually
 allocated.

16/CTP\$BUFLNAME 4 Byte value with the name of the buffer
 allocated.

7.29 Maintenance Buffer Unmap Request Message

00/CTP\$OPCODE 1 Byte, value is 14/CTP\$MBUFUNMREQ.
01/CTP\$REFERENCE 4 Bytes, contains reference number.
05/CTP\$BUFTYPE 1 Byte indicating type of buffer to
 deallocate.

03/CTP\$MAINTBUF Maintenance Data Buffer.

06/CTP\$RESERV6 3 Bytes, must be zero.
09/CTP\$RESERV9 1 Byte, must be zero.
10/CTP\$RESERV10 2 Bytes, must be zero.
12/CTP\$RESERV12 4 Bytes, must be zero.
16/CTP\$BUFLNAME 4 Byte value with the name of the buffer to
release. Must be equal to the name of the
buffer mapped.

7.30 Maintenance Buffer Unmap Response Message

00/CTP\$OPCODE 1 Byte, value is 78/CTP\$MBUFUNMRSP.
01/CTP\$REFERENCE 4 Bytes, contains reference number.
05/CTP\$STATUS 1 Byte indicating type of buffer deallocated,
values are:

00/CTP\$SUCCESS Function Successful.
249/CTP\$NOBUFMAPPED No Buffer Mapped.
253/CTP\$NORESOURCE Resource Failure.
254/CTP\$NONSENSE Request Not Understood.
255/CTP\$NOTIMP Function Not Implemented.

06/CTP\$RESERV6 3 Bytes, must be zero.
09/CTP\$RESERV9 1 Byte, must be zero.
10/CTP\$RESERV10 2 Bytes, must be zero.
12/CTP\$RESERV12 4 Bytes, must be zero.
16/CTP\$BUFLNAME 4 Byte value with the name of the buffer to
release. Must be equal to the name of the
buffer mapped.

7.31 Move Maintenance Buffer Request Message

00/CTP\$OPCODE 1 Byte, value is 15/CTP\$MOVMBUFREQ.
01/CTP\$REFERENCE 4 Bytes, contains reference number.
05/CTP\$DELAY 1 Byte specifying delay prior to action
(~500ms/count).
06/CTP\$REPCOUNT 2 Byte integer containing a repeat count.

08/CTP\$MOVETYPE 1 Byte specifying the type of buffer move to perform:

00/CTP\$MNTMOVEFROM Responder does maint read.
01/CTP\$MNTMOVETO Responder does maint write.

09/CTP\$OTHERNODE 1 Byte containing the number of the other node involved.

10/CTP\$PKTSIZ 1 Byte, contains the base packet size to use for the transfer.

00/CTP\$PS512 Size is 512 bytes.
01/CTP\$PS576 Size is 576 bytes.

11/CTP\$RESERV11 1 Byte, must be zero.

12/CTP\$BUFLNGTH 4 Byte integer containing the size of the move in bytes.

16/CTP\$BUFLNAME 4 Byte value with the local buffer name.

20/CTP\$RESERV20 4 Bytes, must be zero.

24/CTP\$BUFRNAME 4 Byte value with the remote buffer name.

7.32 Move Maintenance Buffer Response Message

00/CTP\$OPCODE 1 Byte, value is 79/CTP\$MOVMBUFRSP.

01/CTP\$REFERENCE 4 Bytes, contains reference number.

05/CTP\$STATUS 1 Byte with the SCA or CTP status of the last move performed. Values are:

00/CTP\$SUCCESS Function Successful.
246/CTP\$NOCONNECT No connection exists.
247/CTP\$PKTSIZFAIL Unable to use specified packet size.
249/CTP\$NOBUFRMAPPED No Buffer Mapped.
251/CTP\$FAILURE Function Failure.
252/CTP\$REFUSE Request refused.
253/CTP\$NORESOURCE Resource Failure.
254/CTP\$NONSENSE Request Not Understood.
255/CTP\$NOTIMP Function Not Implemented.

06/CTP\$ACTCOUNT 2 Byte integer holding the actual number of moves done.

08/CTP\$MOVETYPE 1 Byte specifying the type of buffer move performed:

00/CTP\$MNTMOVEFROM Responder did maint read.
01/CTP\$MNTMOVEETO Responder did maint write.

09/CTP\$OTHERNODE 1 Byte containing the number of the other node involved.

10/CTP\$PKTSIZ 1 Byte, contains the base packet size to used for the transfer. Same as specified in request.

00/CTP\$PS512 Size is 512 bytes.
01/CTP\$PS576 Size is 576 bytes.

11/CTP\$RESERV11 1 Byte, must be zero.

12/CTP\$BUFLNGTH 4 Byte integer containing the size of the move in bytes. Same as specified in request.

16/CTP\$BUFLNAME 4 Byte value with the local buffer name.

20/CTP\$RESERV20 4 Bytes, must be zero.

24/CTP\$BUFRNAME 4 Byte value with the remote buffer name.

7.33 Maintenance State Request Message

00/CTP\$OPCODE 1 Byte, value is 16/CTP\$MSTATEREQ

01/CTP\$REFERENCE 4 Bytes, contains reference number.

05/CTP\$RESERV05 3 Bytes, must be zero.

08/CTP\$GENFUNCT 1 Byte, indicating direction of state change.

00/CTP\$MAINTSTATE Get port into the maintenance enabled state.
01/CTP\$NORMSTATE Get port into the normal operating state.

7.34 Maintenance State Response Message

00/CTP\$OPCODE 1 Byte, value is 80/CTP\$MSTATERSP

01/CTP\$REFERENCE 4 Bytes, contains reference number.

05/CTP\$STATUS 1 Byte indicating status of request:

00/CTP\$SUCCESS	Function Successful, in maintenance
enabled state.	
01/CTP\$GOTO	Will attempt to get into the
maintenance enabled	state.
251/CTP\$FAILURE	Function Failure.
252/CTP\$REFUSE	Request refused.
254/CTP\$NONSENSE	Request Not Understood.
255/CTP\$NOTIMP	Function Not Implemented.

Note -

The maintenance state command is used in conjunction with maintenance type functions. If a node is already in the maintenance state, the response to a request to get into the maintenance enabled state should be returned with success status. This indicates that the node is presently in the maintenance enabled state, or that it was some how able to honor the request without disturbing virtual circuits and connections. HSC50 for example always operates in the maintenance enabled state and therefor would always respond with success status. For nodes that must take their port down, thus closing virtual circuits and connections, the response CTP\$GOTO should be returned. After virtual circuits are established, and the controller connects to the responder, the request can be repeated, and assuming the function worked, a success status would then be returned. Going in the other direction, from the maintenance enabled state to enabled state (or normal operating state) would be accomplished in a similar manner. If the node was able to go to its normal operating state from the maintenance enabled state without disturbing virtual circuits and connections, the response would be success. In the case of HSC50, the response would again be success, because it is in its normal operating state. If virtual circuits are broken, the response would be CTP\$GOTO, indicating that after establishing virtual circuits, the node would be in its normal operating state. The return to normal operating state function would not be used, if the node was reset and started.

The sequence of functions with respect to the maintenance state command is important. A controller wishing to test a maintenance function, must first request the node to get into the maintenance enabled state, before mapping maintenance buffer, and maintenance buffer should be unmapped before going back to the enabled state. This must be done because

maintenance buffers mapped, will (or may be) unmapped when virtual circuits/connections to controllers are lost due to the state change. If all connections to controllers are lost, maintenance buffers, as well as normal data buffers should be unmapped.

APPENDIX A

MULTIPLE CI DROPS ON ONE PROCESSOR

If one processor (or multiprocessor) has more than one port to the same CI star coupler, it will appear as multiple nodes on the CI. For cluster test, each node must be separately addressable.

With respect to CTP operation, such a configuration must appear to be more than one processor. In other words, the responder must be available at each node address (either two different responders or the same one handling both ports individually), and resource requirements must be met for each of the ports that is connected.

APPENDIX B

MULTIPLE CTP CONTROLLERS ACTIVE ON ONE CI

If multiple controllers become active on one CI, the responder in each node will not be responsible for resolving conflicts. The resources the responder is required to supply need not be extended, but may be divided in any manner among the controllers requesting them.

The best method for resolving such conflicts would be to avoid them. If a controller is running in an environment that would support multiple controllers, it should not request actions that may be overridden by other controllers. More elaborate mechanisms for conflict resolution are left as an exercise for the reader.

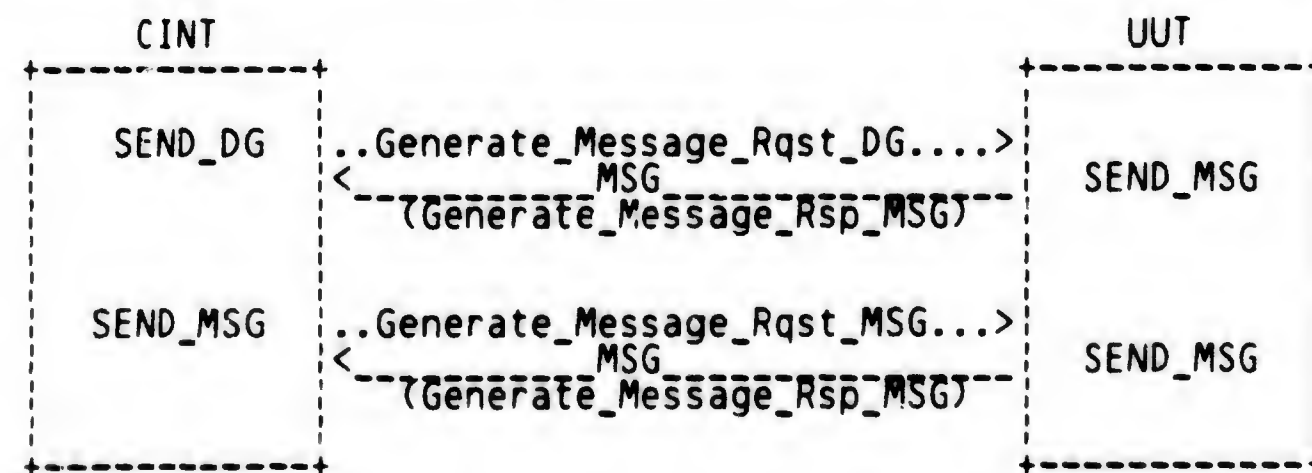
APPENDIX C

CLUSTER TEST SCENARIOS

This appendix presents some examples of CI cluster testing, to illustrate the usage of CTP and SCA in the diagnostic environment. These examples have been drawn from the VAX CI Exerciser and the CI Node Tester programs.

C.1 BASIC SEQUENCED MESSAGE TEST

This test will verify the UUT's ability to transmit and receive MSG packets. The CINT will send a Generate Message Request Datagram to the UUT requesting the UUT to build and send a MSG with a known data pattern. The CINT will verify the data contained in the received message. The CINT will then send a Generate Message Request Message with a known data pattern. The CINT will verify the data contained in the received message.



..... = Cluster Test Protocol request/response

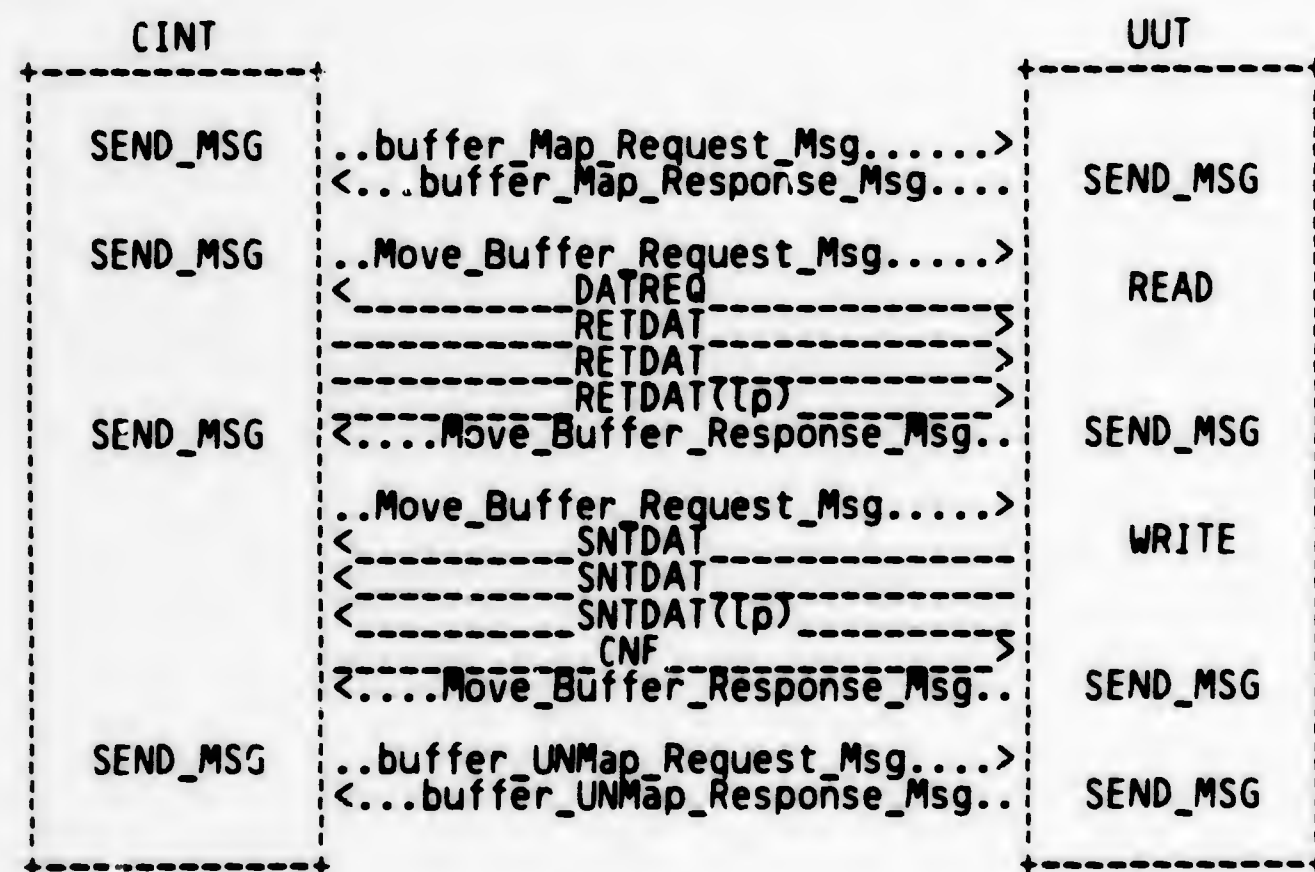
----- = CI Data Link Packet

< or > = direction of transfer, < is from UUT to CINT
> is from CINT to UUT

SEND_DG = VMS System Communication Services operation
or SEND_MSG

C.2 UUT READ FROM CINT BUFFER TEST.

This test will verify the UUT's ability to send a DATREQ packet and receive a RETDAT packet. The CINT will tell the UUT to READ data from a CINT buffer. After the UUT transfers the data, the CINT will tell the UUT to send the contents of the receive buffer used for the READ back to the CINT for checking. The UUT's ability to WRITE to the CINT has been previously tested.



..... = Cluster Test Protocol request/response

----- = CI Data Link Packet

< or > = direction of transfer, < is from UUT to CINT
 > is from CINT to UUT

SEND_MSG = VMS System Communication Services operation
 WRITE
 READ

APPENDIX D MESSAGE OFFSET SUMMARY

00/CTP\$OPCODE	1 Byte with message opcode
01/CTP\$REFERENCE	4 Bytes with reference number
05/CTP\$BUFTYPE	1 Byte indicating type of buffer
05/CTP\$DELAY	1 Byte with delay (~500ms/count)
05/CTP\$RESERV5	4 Bytes of space filler
05/CTP\$RESERV05	3 Bytes of space filler
05/CTP\$STATUS	1 Byte with CTP request status
06/CTP\$ACTCOUNT	2 Bytes with actual action count
06/CTP\$FMASK	32 Bytes containing a mask
06/CTP\$REPCOUNT	2 Bytes containing a repeat count
06/CTP\$RESERV6	3 Bytes of space filler
06/CTP\$RESERV06	2 Bytes of space filler
08/CTP\$EXTEND	1 Byte specifying defaults
08/CTP\$GENFUNCT	1 Byte specifying a function
08/CTP\$MOVETYPE	1 Byte specifying the type of move
09/CTP\$GENCONST	1 Byte with a fill constant
09/CTP\$RESERV9	1 Byte of space filler
: 09/CTP\$OTHERNODE	1 Byte containing a node number
10/CTP\$NOACTFLAG	1 Byte action for no unsolicited activity
: 10/CTP\$PKTSIZ	1 Byte with the base packet size
: 10/CTP\$CNTFLG	1 Byte indicating to keep or release counters
10/CTP\$RESERV10	2 Bytes of space filler
11/CTP\$PKTMULT	1 Byte with the packet size multiple
11/CTP\$CFGPOSTS	1 Byte containing path 0 status
: 11/CTP\$RESERV11	1 Bytes of space filler
12/CTP\$GENLENGTH	2 Bytes indicating generated length
12/CTP\$CFGPOSTS	1 Byte containing path 1 status
12/CTP\$BUFLNGTH	4 Bytes with the size of a buffer
12/CTP\$CNTRPOACK	4 Bytes containing ACKs on path 0
12/CTP\$STARTADR	4 Bytes containing optional start address
: 12/CTP\$RESERV12	4 Bytes of space filler
14/CTP\$IMAGEDATA	N Bytes containing image data
14/CTP\$GENDATA	N Bytes containing generated data

MESSAGE OFFSET SUMMARY

Page D-2
12 Apr 83

SEQ 0082

16/CTP\$BUFLNAME	4 Bytes with local buffer name
16/CTP\$CNTRPONAK	4 Bytes containing NAKs on path 0
20/CTP\$BUFLOFSET	4 Bytes with local buffer offset
20/CTP\$CNTRPONRSP	4 Bytes containing path 0 no responses
20/CTP\$RESERV20	4 Bytes of space filler
24/CTP\$BUFRNAME	4 Bytes with remote buffer name
24/CTP\$CNTRP1ACK	4 Bytes containing ACKs on path 1
28/CTP\$BUFROFSET	4 Bytes with remote buffer offset
28/CTP\$CNTRP1NAK	4 Bytes containing NAKs on path 1
32/CTP\$CNTRP1NRSP	4 Bytes containing total NORSPs in path 1
36/CTP\$CNTRDISCDG	4 Bytes containing total discarded datagrams

APPENDIX E OPCODE SUMMARY

00/CTPS\$FUNCTREQ	Responder Function Set Request
01/CTPS\$BUFMAPREQ	Buffer Map Request
02/CTPS\$BUFUNMREQ	Buffer Unmap Request
03/CTPS\$MOVBUFREQ	Move Buffer Request
04/CTPS\$GENMSGREQ	Generate Message Request
05/CTPS\$GENDGREQ	Generate Datagram Request
06/CTPS\$GENRSTREQ	Generate Reset Request
07/CTPS\$GENSTRREQ	Generate Start Request
08/CTPS\$NOACTREQ	Stop/Start Unsolicited Activity Request
09/CTPS\$CONFIGREQ	Configuration Data Request
10/CTPS\$COUNTSREQ	Counter Read Request
11/CTPS\$CONNECTREQ	Third Party Connect Request
12/CTPS\$FINISHREQ	Listener Disconnect Request
13/CTPS\$MBUFMAPREQ	Maintenance Buffer Map Request
14/CTPS\$MBUFUNMREQ	Maintenance Buffer Unmap Request
15/CTPS\$MOVMBUFRREQ	Move Maintenance Buffer Request
16/CTPS\$MSTATEREQ	Maintenance State Request

64/CTPS\$FUNCTRSP	Responder Function Set Response
65/CTPS\$BUFMAPRSP	Buffer Map Response
66/CTPS\$BUFUNMRSP	Buffer Unmap Response
67/CTPS\$MOVBUFRSP	Move Buffer Response
68/CTPS\$GENMSGRSP	Generate Message Response
69/CTPS\$GENDGRRSP	Generate Datagram Response
70/CTPS\$GENRSTRSP	Generate Reset Response
71/CTPS\$GENSTRRSP	Generate Start Response
72/CTPS\$NOACTRSP	Stop/Start Unsolicited Activity Response
73/CTPS\$CONFIGRSP	Configuration Data Response
74/CTPS\$COUNTSRSP	Counter Read Response
75/CTPS\$CONNECTRSP	Third Party Connect Response
76/CTPS\$FINISHRSP	Listener Disconnect Response
77/CTPS\$MBUFMAPRSP	Maintenance Buffer Map Response
78/CTPS\$MBUFUNMRSP	Maintenance Buffer Unmap Response
79/CTPS\$MOVMBUFRSP	Move Maintenance Buffer Response
80/CTPS\$MSTATERSP	Maintenance State Response

255/CTPS\$OPEXPAND	Expanded Opcode Introducer
--------------------	----------------------------

APPENDIX F

CTP STATUS AND VALUES SUMMARY

00/CTP\$SUCCESS	Function Successful
00/CTP\$DEFAULTADR	Use node's default address
00/CTP\$NOACTON	Resume normal unsolicited activity
00/CTP\$GENFFILL	Fill buffer with constant
00/CTP\$MOVEFROM	Responder does a read
00/CTP\$MNTMOVEFROM	Responder does maint read
00/CTP\$UNCONDRST	Forced reset
00/CTP\$PS512	Packet size is 512 bytes
00/CTP\$PATHBAD	Path is marked bad
00/CTP\$MAINTSTATE	Change to maintenance state
00/CTP\$RELCNT	Release control of counters
01/CTP\$GOTO	Will attempt to get to maintenance state
01/CTP\$NOACTOFF	Stop normal unsolicited activity
01/CTP\$SPECIFADR	Use the specified address
01/CTP\$REALBUF	Real Memory Buffer
01/CTP\$GENFBPAIR	Fill w/byte pair numbers
01/CTP\$MOVETO	Responder does a write
01/CTP\$MNTMOVETO	Responder does maint write
01/CTP\$CONDRST	Conditionally reset
01/CTP\$MAINTSTATE	Go to maintenance enabled state
01/CTP\$PS576	Packet size is 576 bytes
01/CTP\$PATHGOOD	Path is marked good
01/CTP\$NORMSTATE	Go to normal operating state
01/CTP\$KEEPCNT	Keep control of counters
02/CTP\$FAKEBUF	'Black Hole' Buffer
02/CTP\$GENFNODE	Fill w/responder node number
03/CTP\$MAINTBUF	Maintenance Data Buffer
03/CTP\$GENFIMAGE	Fill w/data in this packet
245/CTP\$CONTRANS	Connection is in transition state
246/CTP\$NOCONNECT	No connection exists
247/CTP\$PKTSIZFAIL	Unable to use specified packet size
248/CTP\$NODEUNKNOWN	Node Unknown To SCA
249/CTP\$NOBUFMAAPPED	No Buffer Mapped
250/CTP\$BUFLIMIT	Buffer Map Limit Exceeded
251/CTP\$FAILURE	Function failure
252/CTP\$REFUSE	Request refused

CTP STATUS AND VALUES SUMMARY

Page F-2
12 Apr 83

SEQ 0085

253/CTP\$NORESOURCE	Resource Failure
254/CTP\$NONSENSE	Request Not Understood
255/CTP\$NOTIMP	Function Not Implemented

CODE: MAINDEC-10-DFCIB
TITLE: Decsystem 2080 Computer Interconnect Exerciser (DFCIB)
VERSION: 0.1
DATE: October 1984
REASON: Preliminary release of this program for KLIPA (IPA20).

PRODUCT CODE: AH-T734A-DD
DIAGNOSTIC CODE: DFCIB
PRODUCT NAME: DFCIBB0 CI20 CTP Exerciser
VERSION: 2
DATE RELEASED: August 21, 1985
UPDATE AUTHOR: Gregory A. Scott
REASON: Install exerciser (TST99) (Ed Tulloch)
Repair bugs in exerciser and other tests
Update to TOPS-10 operation using GLXLIB
Update and enhance documentation

DOCUMENT

DFCIB LST

COPYRIGHT 1985
DIGITAL EQUIPMENT CORPORATION
MAYNARD, MASS. 01754

TABLE OF CONTENTS

41	
84	GLXLIB Symbols
4	Essentially GLXINI for -10 and -20 both /GAS
19	User referenced entry vector
47	I%INIT - Initialize the runtime system
77	ACQLIB - Acquire the run-time system
118	CHKLIB - Routine to test if library is in place
148	TRYSEG - Routine to fetch runtime system
85	Storage
120	Subroutines
992	Perform SCS Functions
1098	Literals
50	Footprint Definitions
65	Build CTP Packets
632	Send Message/Datagram
702	Read Message/Datagram
838	Requeue Message/Datagram Buffer
876	Subroutines
953	Field Byte Pointers
984	Storage
1049	Literals
42	Listen

TABLE OF CONTENTS

81	Accept Request
114	Queue Listening Buffers
175	Return Message
249	Request Data
398	Literals
154	GLXLIB Interface
237	General Storage
289	Parse Tables
538	Initial Start Up
642	Top Level Parser
673	Clear Command
703	Set Command
825	Select Command
876	Deselect Command
915	HELP Command
1075	RUN COMMAND
1267	Show Command
1441	TAKE Command
1537	Quit Command
1554	Send Command
1601	Connect/Disconnect Commands
1614	Wait Command

TABLE OF CONTENTS

2011	Read Command
2041	Requeue Command
2073	Type Command
2136	Map-Buffer/Unmap-Buffer Commands
2162	PAUSE/LISTEN/ACCEPT Commands
2193	DDT Command
2247	Request-Data Command
2265	Get First Node/Next Node
2308	Logging Subroutines
2385	Parsing Subroutines
2467	ERRHAN - Produce Error Messages
2517	Run Time Information
2595	Literals
123	Derinitions
174	Scripts
244	TST01 Connect/Disconnect Test
286	TST02 - Transmit Function Set Request/Receive Response
360	TST10 - Basic Message Test
448	TST11 Message Test - 4 bytes each of 4 data types
809	TST12 Message Test - 4 bytes each of 4 data types (repeat=3)
1693	TST13 - Message Test - 4 bytes of 4 data types (delay of 2 sec)
2050	TST14 - Message Test - 1 to max bytes each of 4 data types

TABLE OF CONTENTS

2540	TST30 - Basic Datagram Test
2643	TST31 - Datagram 4 Bytes Each Of 4 Data Types
3088	TST32 - Datagram 4 Bytes Each Of 4 Data Types (Repeat=3)
4248	TST33 - Datagram 4 Bytes Each Of 4 Data Types (Delay = 2 Sec)
4695	TST40 - Datagram 4 Bytes Each Of 4 Data Types (Delay = 2 Sec)
4889	TST50 - Buffer Map and Unmap Test
5024	TST51 - Buffer Map, Controller Read, Buffer Unmap Test
5616	TST52 - Buffer Map, Buffer Move (Responder Write), Buffer Unmap
5851	TST53 - Buffer Move (Responder Write, Delay=2 Sec)
6084	TST54 - Buffer Move (Responder Write, Repeat=3) Test
6321	TST55 - Buffer Move (Responder Write, Variable Length) Test
6581	TST56 - Buffer Move (Responder Write, Packet Size=1) Test
6817	TST57 - Buffer Move (Responder Write, Packet Multiple) Test
7065	TST58 - Buffer Move (Exerciser Write, Responder Write) Test
7415	TST59 - Buffer Move (Exerciser Write, Responder Write) Test
7734	TST80 - Read Counter Test
7885	RCVRM - RECEIVE RESPONSE MESSAGE.
7938	XMTREQ - TRANSMIT CTP REQUEST
7973	CONNCT - Connect to node
8004	TRSPNS - Test Response Message Opcode
8019	TSTAT - Test Response Message Status
8034	TSTACT - Test for act count equal to expected

TABLE OF CONTENTS

8055	WAITDA - Wait for datagram available subroutine
8072	DMAEFR - Wait for DMA Transfer Complete Subroutine
8089	TEVPND - Wait for Event Pending Flag
8107	CONACC - Wait For Connection Accepted
8125	CONOPN - Wait for Connection Open
8143	CONREJ - Wait for Connection Rejected
8158	CONCLO - Wait for Connection Closed
8176	SNDCMP - Wait for Message/Datagram Send Complete
8194	MSCAVL - Wait for Message Available
8212	DATAVL - Wait for Datagram Available
8230	RNVERF - Reference Number Verify Routine
8244	TRNODE - Print Node Number Testing
8273	Literals and Storage
151	Exerciser Code
301	Initialization Subroutines
374	Selection Subroutines
496	Deselection Subroutines
512	Send Message Subroutines
638	Transmit Request
655	Send Datagrams
800	Utility Subroutines
1061	XRCVRM/XRCVRD - RECEIVE RESPONSE MESSAGE/DATAGRAM

TABLE OF CONTENTS

1145	Map/Move Buffer Subroutines
1788	Utility Routines
2336	Exerciser Storage
2492	Literals

41

Macros

84

GLXLIB Symbols

4

Essentially GLXINI for -10 and -20 both /GAS

19

User referenced entry vector

47

IXINIT - Initialize the runtime system

77

ACQLIB - Acquire the run-time system

118

CHKLIB - Routine to test if library is in place

148

TRYSEG - Routine to fetch runtime system

85

Storage

120

Subroutines

122

191

226


```
271 *****
    *****
325 *****
    *****
367 *****
    *****
375 *****
    *****
392 *****
    *****
421 *****
    *****
476 *****
    *****
498 *****
    *****
518 *****
    *****
653 *****
    *****
700 *****
    *****
751 *****
    *****
765 *****
773  Call this routine via '$CALL OTIME'
    *****
796 *****
    *****
840 *****
    *****
859 *****
    *****
```

883 *****

971 *****

992 *****
Perform SCS Functions

1098 *****
Literals

50 *****
Footprint Definitions

65 *****
Build CTP Packets

632 *****
Send Message/Datagram

702 *****
Read Message/Datagram

838 *****
Requeue Message/Datagram Buffer

876 *****
Subroutines

953 *****
Field Byte Pointers

```
*****
984  Storage
*****

1049 Literals
*****

42  Listen
*****

44  *****
    *****

81  Accept Request
*****

83  *****
    *****

114 Queue Listening Buffers
*****

116 *****
    *****

154 *****
    *****

175 Return Message
*****

177 *****
    *****
```

249

Request Data

398

Literals

154

GLXLIB Interface

237

General Storage

289

Parse Tables

538

Initial Start Up

642

Top Level Parser

673

Clear Command

703

Set Command

825

Select Command

```
*****
876  Deselect Command
*****

915  HELP Command
*****

1075 RUN COMMAND
*****

1267 Show Command
*****

1441 TAKE Command
*****

1537 Quit Command
*****

1554 Send Command
*****

1601 Connect/Disconnect Commands
*****

1614 Wait Command
*****

2011 Read Command
*****
```


2041 *****
Requeue Command

2073 *****
Type Command

2136 *****
Map-Buffer/Unmap-Buffer Commands

2162 *****
PAUSE/LISTEN/ACCEPT Commands

2193 *****
DDT Command

2247 *****
Request-Data Command

2265 *****
Get First Node/Next Node

2308 *****
Logging Subroutines

2385 *****
Parsing Subroutines

2467 *****
ERRHAN - Produce Error Messages

```
*****
2517 Run Time Information
*****

*****
2595 Literals
*****

*****
123 Definitions
*****

*****
174 Scripts
*****

*****
244 TST01 Connect/Disconnect Test
*****

246 *****
    TST01 Connect then Disconnect to selected node.
    *****

*****
286 TST02 - Transmit Function Set Request/Receive Response
*****

288 *****
    TST02 - TRANSMIT FUNCTION SET REQUEST/RECEIVE RESPONSE *
    *****

*****
360 TST10 - Basic Message Test
*****

362 *****
    TST10 - BASIC MESSAGE TEST. THIS TEST WILL SEND A MESSAGE REQUEST OF
    *****
```

448

TST11 Message Test - 4 bytes each of 4 data types

450

TST11 - MESSAGE TEST. THIS TEST WILL SEND A MESSAGE REQUEST OF

542

TST11A - MESSAGE TEST. THIS TEST WILL SEND A MESSAGE REQUEST OF

627

TST11B - MESSAGE TEST. THIS TEST WILL SEND A MESSAGE REQUEST OF

695

TST11C - MESSAGE TEST. THIS TEST WILL SEND A MESSAGE REQUEST OF

809

TST12 Message Test - 4 bytes each of 4 data types (repeat=3)

811

TST12 - MESSAGE TEST. THIS TEST WILL SEND A MESSAGE REQUEST OF

1004

TST12A - MESSAGE TEST. THIS TEST WILL SEND A MESSAGE REQUEST OF

1235

TST12B - MESSAGE TEST. THIS TEST WILL SEND A MESSAGE REQUEST OF

1414

TST12C - MESSAGE TEST. THIS TEST WILL SEND A MESSAGE REQUEST OF

1693

TST13 - Message Test - 4 bytes of 4 data types (delay of 2 sec)

1695

TST13 - MESSAGE TEST. THIS TEST WILL SEND A MESSAGE REQUEST OF

1786

TST13A - MESSAGE TEST. THIS TEST WILL SEND A MESSAGE REQUEST OF

1876

TST13B - MESSAGE TEST. THIS TEST WILL SEND A MESSAGE REQUEST OF DATA=

1940

TST13C - MESSAGE TEST. THIS TEST WILL SEND A MESSAGE REQUEST OF DATA=IMAGE

2050

TST14 - Message Test - 1 to max bytes each of 4 data types

2052

TST14 - MESSAGE TEST. THIS TEST WILL SEND A GROUP OF MESSAGE REQUESTS OF

2191

TST14A - MESSAGE TEST. THIS TEST WILL SEND A GROUP OF MESSAGE REQUESTS OF

2307

TST14L - MESSAGE TEST. THIS TEST WILL SEND A GROUP OF MESSAGE REQUESTS

2400

TST14C - MESSAGE TEST. THIS TEST WILL SEND A GROUP OF MESSAGE REQUESTS OF

2540 TST30 - Basic Datagram Test

2542 *****
TST30 - BASIC DATAGRAM TEST. THIS TEST WILL SEND A DATAGRAM REQUEST OF

2643 *****
TST31 - Datagram 4 Bytes Each Of 4 Data Types

2645 *****
TST31 - DATAGRAM TEST. THIS TEST WILL SEND A DATAGRAM REQUEST OF

2760 *****
TST31B - DATAGRAM TEST. THIS TEST WILL SEND A DATAGRAM REQUEST OF

2863 *****
TST31C - DATAGRAM TEST. THIS TEST WILL SEND A DATAGRAM REQUEST OF

2972 *****
TST31D - DATAGRAM TEST. THIS TEST WILL SEND A DATAGRAM REQUEST OF

3088 *****
TST32 - Datagram 4 Bytes Each Of 4 Data Types (Repeat=3)

3090 *****
TST32 - DATAGRAM TEST. THIS TEST WILL SEND A DATAGRAM REQUEST OF

3381 *****
TST32B - DATAGRAM TEST. THIS TEST WILL SEND A DATAGRAM REQUEST OF

3662 *****
TST32C - DATAGRAM TEST. THIS TEST WILL SEND A DATAGRAM REQUEST OF

3949 *****
TST32D - DATAGRAM TEST. THIS TEST WILL SEND A DATAGRAM REQUEST OF

4248 *****
TST33 - Datagram 4 Bytes Each Of 4 Data Types (Delay = 2 Sec)

4250 *****
TST33 - DATAGRAM TEST. THIS TEST WILL SEND A DATAGRAM REQUEST OF

4363 *****
TST33B - DATAGRAM TEST. THIS TEST WILL SEND A DATAGRAM REQUEST OF

4463 *****
TST33C - DATAGRAM TEST. THIS TEST WILL SEND A DATAGRAM REQUEST OF DATA=

4574 *****
TST33D - DATAGRAM TEST. THIS TEST WILL SEND A DATAGRAM REQUEST OF DATA=

4695 *****
TST40 - Datagram 4 Bytes Each Of 4 Data Types (Delay = 2 Sec)

4697 *****
TST40 - DATAGRAM TEST. THIS TEST WILL SEND A DATAGRAM REQUEST OF

4889 *****
TST50 - Buffer Map and Unmap Test

4891 *****
TST50 - BUFFER MAP(CTP=01) AND UNMAP(CTP=02) TEST

5024 *****
TST51 - Buffer Map, Controller Read, Buffer Unmap Test

5026 *****
TST51 - BUFFER MAP,CONTROLLER READ (C=.SSREQ),BUFFER UNMAP TEST

5616

TST52 - Buffer Map, Buffer Move (Responder Write), Buffer Unmap

5618

TST52 - BUFFER MAP,BUFFER MOVE(RESPONDER WRITE),BUFFER UNMAP TEST

5851

TST53 - Buffer Move (Responder Write, Delay=2 Sec)

5853

6084

TST54 - Buffer Move (Responder Write, Repeat=3) Test

6086

TST54 - BUFFER MOVE(RESPONDER WRITE -REPEAT COUNT=3) TEST

6321

TST55 - Buffer Move (Responder Write, Variable Length) Test

6323

6581

TST56 - Buffer Move (Responder Write, Packet Size=1) Test

6583

6817

TST57 - Buffer Move (Responder Write, Packet Multiple) Test

6819


```
*****
7065  TST58 - Buffer Move (Exerciser Write, Responder Write) Test
*****

7067  *****
      TST58 BUFFER MOVE(EXERCISER WRITE-RESPONDER WRITE) TEST
      *****

7415  *****
      TST59 - Buffer Move (Exerciser Write, Responder Write) Test
      *****

7417  *****
      TST59 BUFFER MOVE(EXERCISER WRITE-RESPONDER WRITE) TEST
      *****

7734  *****
      TST80 - Read Counter Test
      *****

7736  *****
      TST80 - Read Counter Test.
      *****

7885  *****
      RCVRM - RECEIVE RESPONSE MESSAGE.
      *****

7889  This subroutine returns +1 SUCCESSFUL TRANSMIT OF REQUEST.

7938  *****
      XMREQ - TRANSMIT CTP REQUEST
      *****

7942  This subroutine returns +1 SUCCESSFUL TRANSMIT OF REQUEST.

7973  *****
      CONNCT - Connect to node
      *****

7976  This subroutine returns +1 GOOD CONNECTION(ACCEPTED AND OPEN)
```

8004 *****
TRSPNS - Test Response Message Opcode

8008 This subroutine returns +1 GOOD COMPARE

8019 *****
TSTAT - Test Response Message Status

8023 This subroutine returns +1 GOOD COMPARE

8034 *****
TSTACT - Test for act count equal to expected

8037 TSTACT Subroutine
THIS SUBROUTINE CHECKS A TWO BYTE ACTUAL COUNT FIELD.
 16 23 24 31
 +-----+-----+-----+
 ! LEAST SIGNIFICANT BYTE! MOST SIGNIFICANT BYTE !
 +-----+-----+-----+

This subroutine always returns: +1 ON GOOD COMPARISON.

8055 *****
WAITDA - Wait for datagram available subroutine

8058 WAITDA Subroutine
This subroutine returns +1 if datagram available flag did set.
This subroutine returns +2 if datagram available flag did not set.

8072 *****
DMAXFR - Wait for DMA Transfer Complete Subroutine

8075 DMAXFR Subroutine
This subroutine returns +1 if DMA transfer complete flag did set.
This subroutine returns +2 if DMA transfer complete flag did not set.


```
*****
8089  TEVPND - Wait for Event Pending Flag
*****

8092  TEVPND Subroutine
      This subroutine returns :      +1 ON GOOD COMPARE

*****
8107  CONACC - Wait For Connection Accepted
*****

8110  CONACC Subroutine
      This subroutine returns :      +1 ON GOOD COMPARE

*****
8125  CONOPN - Wait for Connection Open
*****

8128  CONOPN Subroutine
      This subroutine returns :      +1 ON GOOD COMPARE

*****
8143  CONREJ - Wait for Connection Rejected
*****

8146  CONREJ Subroutine
      This subroutine returns :      +1 ON GOOD COMPARE

*****
8158  CONCLO - Wait for Connection Closed
*****

8161  CONCLO Subroutine
      This subroutine returns :      +1 ON GOOD COMPARE

*****
8176  SNDCMP - Wait for Message/Datagram Send Complete
*****

8179  SNDCMP Subroutine
      This subroutine returns :      +1 ON GOOD COMPARE
```


8194 *****
MSGAVL - Wait for Message Available

8197 MSGAVL Subroutine
This subroutine returns : +1 ON GOOD COMPARE

8212 *****
DATAVL - Wait for Datagram Available

8215 DATAVL Subroutine
This subroutine returns : +1 ON GOOD COMPARE

8230 *****
RNVERF - Reference Number Verify Routine

8244 *****
TRNODE - Print Node Number Testing

8273 *****
Literals and Storage

151 *****
Exerciser Code

301 *****
Initialization Subroutines

374 *****
Selection Subroutines

496 *****
Deselection Subroutines

512

Send Message Subroutines

638

Transmit Request

655

Send Datagrams

800

Utility Subroutines

1061

XRCVRM/XRCVRD - RECEIVE RESPONSE MESSAGE/DATAGRAM

1065 This subroutine returns +1 SUCCESSFUL TRANSMIT OF REQUEST.

1076 This subroutine returns +1 SUCCESSFUL TRANSMIT OF REQUEST.

1145

Map/Move Buffer Subroutines

1788

Utility Routines

1895 SSRCD - Return Connection Data (about a remote node)

1953 SSSCON - Request a connection with another node on the CI.

2008 SSEVT - Get Entry from Event Queue

2045 SSDIS - Disconnect from a remote node

2079 SSSMG - Send a message to a remote node.

2115 SSQRM - Queue Message Buffers - receive a message from a DUP server.

2144 SSCRM - Cancel (dequeue) Receive Message

2174 SSCSP - Connect State Poll - status of connection

2207 SSSTS - Status Info on Connection (Fast form of .SSCSP)

2240 SSRMG - Receive Message

2271 SSGDE - Get Entry from Data Queue

2297 *****

2336 *****
Exerciser Storage

2492 *****
Literals

!DFCIB.HLP Online help file for DFCIB version 2(16) updated 22-Aug-85

SEQ 0113

!Each DFCIB command has an entry in this file. The entries must be
!alphabetically ordered and must be preceded with an '*' in column 1.
!Comments have a '!' in column 1. Additional entries may be added to
!this help file, but care must be taken to follow the above.

Help for a specific command may be obtained with the HELP command as follows:

HELP command-name

For instance if you wish additional information about the TAKE command you would type 'HELP TAKE'.

*ACCEPT

The ACCEPT command is used only for writing your own tests. It causes a connection to be accepted from another exerciser program.

*BUILD

BUILD is a command used only for writing your own tests. It allows the building of CTP packets for later transmission using the SEND command.

*CLEAR

The CLEAR command is the complement of the SET command. For more information, enter one of the following:

HELP CLEAR LOGGING
HELP CLEAR SWITCH

*CLEAR LOGGING

The CLEAR LOGGING command is used to close the log file, which is opened by the SET LOGGING command. The log file receives a copy of all output directed to the terminal.

*CLEAR SWITCH

The CLEAR SWITCH command is used to clear one or more of the switches. It is the complement of the SET SWITCH command. The current switch settings are displayed with the SHOW SWITCHES command. The switches are:

BELL	Ring the terminal bell on errors.
HALT	Stop testing when any error is detected.
PALL	Print all bytes in buffer (TYPE command).
TRACE	Print trace messages as each test is executed.

*CONNECT

The CONNECT command is used only for writing your own tests. It causes a connection to be made to the last node specified in the SELECT command.

*DDT

The DDT command is used only for debugging of the diagnostic. It causes execution to be transferred to DDT if it is loaded. If DDT is not loaded it attempts to load it.

*DESELECT

The DESELECT command is used to disable selection of a node for testing. It is followed by one or more decimal node numbers, separated by commas. To see what nodes are selected, use the SHOW SELECTED-NODES command.

*DISCONNECT

The DISCONNECT command is used only for writing your own tests. It causes the connection created by the CONNECT command to be broken.

*DESELECT

The DESELECT NODE command is the complement of the SELECT NODE command. It is followed by one or more decimal node numbers separated by commas. To see what nodes are selected, use the SHOW SELECTED-NODES command.

*HELP

The HELP command followed by question mark (?) will provide the user with a list of the entries in the online help file. HELP followed by the subject will provide the user with a short message on the selected subject. Help with no subject specified will provide the user with a help message on using the HELP command.

*LISTEN

The LISTEN command is used only for writing your own tests. It causes the program to listen for a CTP connection from any node.

*MAP-BUFFER

The MAP-BUFFER command is used only for writing your own tests. It causes a map of an internal buffer of the number of bytes specified in the command.

*PAUSE

The PAUSE command is normally used only for writing your own tests. It causes the program to sleep for the number of seconds furnished in the command.

*QUIT

The QUIT command causes the diagnostic to exit.

*READ

The READ command is used only for writing your own tests. It is followed by a keyword, DATAGRAM or MESSAGE, indicating the read action.

*REQUEST-DATA

The REQUEST-DATA command is normally used only for writing your own tests. It is used to request DMA transfer data from either a RESPONDER-NODE or an EXERCISER-NODE, as specified in the command.

*REQUEUE

The REQUEUE command is normally used only for writing your own tests. It causes a buffer used by the READ command to be freed up for later use.

*RETURN-MESSAGE

The RETURN-MESSAGE command is used only for writing your own tests. It causes the diagnostic to return a message as a responder program would.

*RUN

The RUN command is used to start the execution of a test or group of tests. The format of the RUN command is:

```
RUN /PASSES:n test/ITERATIONS:n , test/ITERATIONS:n
```

The optional /PASSES switch must be specified before any test name, and specifies how many passes of the test to run. The default number of passes if /PASSES isn't specified is 1.

A test name or test script come next, separated by commas. Each test may be execute a number of times using the optional /ITERATIONS switch. The default number of iterations is 1, except for the EXERCISER and TST99, for which the default iteration count is 100.

A RUN command with no arguments is used to rerun the last successfully entered RUN command. The tests are stored in the "run table", which can be displayed using the SHOW RUN-TABLE command.

An example run command would be

```
DFCIB>RUN /PASSES:100 BASIC/ITER:10, MESSAGE/ITER:5, COUNTER
```

This would run 100 passes, where each pass would consist of 10 iterations of the Basic Connect/Disconnect Tests, 5 iterations of the Message Tests, followed by one iteration of the Counter Test.

During the execution of a test, you can get status of the test or abort testing by using the "run time commands". The run time command options are:

```
H   For this message
S   To get current test status
T   To toggle the TRACE switch
esc Escape to abort testing
```

For more help with the scripts, type HELP followed by a script name, one of the following:

ALL-TESTS	BASIC-TESTS	BUFFER-TESTS	COUNTER-TEST
DATA-TESTS	DEFAULT	EXERCISER	MESSAGE-TESTS

For more help with the tests, type HELP followed by a test name, one

of the following:

TST01	TST02	TST10	TST11	TST12	TST13	TST14	TST30
TST31	TST32	TST33	TST40	TST50	TST51	TST52	TST53
TST54	TST55	TST56	TST57	TST58	TST59	TST80	TST99

*SELECT

The SELECT command is used to select a node for CTP testing. It is followed by node numbers separated by commas. The selected nodes are displayed with the SHOW SELECTED-NODES command.

*SET

The SET command is used to activate various program options. For more information, type one of the following:

```
HELP SET LOGGING
HELP SET PATH-SELECTION
HELP SET SWITCH
HELP SET TIME-OUT
HELP SET TYPE-OUT-LINE-LIMIT
```

*SET LOGGING

The SET LOGGING command is followed by a filename into which all terminal output is placed. The default filename is DSK:DFCIB.LOG. To close the log file, either exit the diagnostic with the QUIT command or use the CLEAR LOGGING command.

*SET PATH-SELECTION

The SET PATH-SELECTION command is used to select either Path A, Path B, or automatic path selection for all CTP packets sent over the CI. The default is to use automatic path selection, which will let the monitor pick the path to use, causing CTP packets to be split more or less evenly across both paths.

*SET SWITCH

The SET SWITCH command is used to set one or more of the switches. It is the complement of the CLEAR SWITCH command. The current switch settings are displayed with the SHOW SWITCHES command. The switches are:

BELL	Ring the terminal bell on errors.
HALT	Stop testing when any error is detected.
PALL	Print all bytes in buffer (TYPE command).
TRACE	Print trace messages as each test is executed.

*SET TIME-OUT

The SET TIME-OUT command is followed by the maximum number of seconds to wait for any event to complete. The default time out is 5 seconds. The SHOW TIME-OUT command is used to display the current time out value.

*SET TYPE-OUT-LINE-LIMIT

The SET TYPE-OUT-LINE-LIMIT command is used to set the number of lines

that are displayed with the TYPE command. The TYPE command is only used for writing your own tests.

SEQ 0117

*SHOW

The SHOW command is used to display various program settings. For more information type HELP SHOW followed by one of:

ALL	CI-CONFIGURATION
COUNTERS	EVENT-QUEUE
ID-OF-CONNECTED-NODE	LOCAL-NODE-NUMBER
MAPPED-BUFFER-NAME	MINIMUM-BUFFER-SIZES
PATH-STATUS-OF-NODE	POLL-STATUS-OF-CONNECTED-NODE
RUN-TABLE	SELECTED-NODES
STATUS-OF-CONNECTED-NODE	SWITCHES
TIME-OUT	

*SHOW ALL-PROGRAM-PARAMETERS

The SHOW ALL-PROGRAM-PARAMETERS command does the equivalent of the following SHOW commands:

- LOCAL-NODE-NUMBER
- PATH-SELECTION
- SELECTED-NODES
- TIME-OUT
- RUN-TABLE
- SWITCHES

The other SHOW commands have to do with writing your own test or getting status of this node or other nodes on the CI. For more information, type HELP SHOW followed by one of the commands.

*SHOW CI-CONFIGURATION

The SHOW CI-CONFIGURATION command will request connect data from all other nodes on the CI and display various node-specific data such as software type and versions, buffer sizes, and so on.

*SHOW COUNTERS

The SHOW COUNTERS function will cause the CI port counters to be read and displayed for this node on the CI.

*SHOW EVENT-QUEUE

The SHOW EVENT-QUEUE command is used only when writing your own tests and shows the status of the event queue for the connection made with the ACCEPT or CONNECT commands.

*SHOW ID-OF-CONNECTED-NODE

The SHOW ID-OF-CONNECTED-NODE command is used only for writing your own tests and shows the connect ID of the node connected with the CONNECT or LISTEN commands.

*SHOW LOCAL-NODE-NUMBER

The SHOW LOCAL-NODE-NUMBER shows the CI node number assigned to the system the diagnostic is running on.

***SHOW MAPPED-BUFFER-NAME**

The SHOW MAPPED-BUFFER-NAME command is used only for writing your own tests and is used to display the buffer name returned by the MAP-BUFFER command.

***SHOW MINIMUM-BUFFER-SIZES**

The SHOW MINIMUM-BUFFER-SIZES command shows what monitor determines is the minimum buffer sizes for datagram and messages.

***SHOW PATH-SELECTION**

The SHOW PATH-SELECTION command will display what paths have been selected with the SET PATH-SELECTION command.

***SHOW PATH-STATUS-OF-NODE**

The SHOW PATH-STATUS-OF-NODE command, followed by a node number, shows what monitor determines is the current status of paths to that node.

***SHOW POLL-STATUS-OF-CONNECTED-NODE**

The SHOW POLL-STATUS-OF-CONNECTED-NODE command is used only for writing your own tests. It shows what monitor knows as the poll status of the currently connected node.

***SHOW RUN-TABLE**

The SHOW RUN-TABLE command is used to display what tests or scripts have been run with the last RUN command. This list of tests/scripts will be used if another RUN command is given with no tests/scripts specified. Type HFLP RUN for more details.

***SHOW SELECTED-NODES**

The SHOW SELECTED-NODES command displays which nodes on the CI have been selected for testing. Nodes are selected for testing by using the SELECT NODE command and are removed from testing by the DESELECT NODE command.

***SHOW STATUS-OF-CONNECTED-NODE**

The SHOW STATUS-OF-CONNECTED-NODE command is used for writing your own tests. It displays the status of the connection made with the CONNECT or LISTEN commands.

***SHOW SWITCHES**

The SHOW SWITCH command is used to display the switch settings. The SET SWITCH command is used to set one or more of the switches. The CLEAR SWITCH command is used to clear switches. The switches are:

BELL	Ring the terminal bell on errors.
HALT	Stop testing when any error is detected.
PALL	Print all bytes in buffer (TYPE command).
TRACE	Print trace messages as each test is executed.

***SHOW TIME-OUT**

The SHOW TIME-OUT command is used to display the time in seconds to wait for any event to complete. The default time out is 5 seconds. The SET TIME-OUT command is used to change the current time out value.

*TAKE

The TAKE command is used to cause a file of commands to be passed to the program. The default filename is DSK:DFCIB.CMD. For example if the following commands were in a file called TEST15.CMD:

```
SELECT NODE 15
SET LOGGING TEST15.LOG
RUN/PASS:3 BUFFER-TESTS/ITER:10,COUNTER-TEST
QUIT
```

The command TAKE TEST15 would cause node 15 to be selected, a log file called TEST15.LOG be opened, three passes of the specified tests to be run, and the program to exit.

*TYPE

The TYPE command is used for writing your own tests. It is followed by one of the following to display the contents of program buffers:

CTPPKT	DBUF1	DBUF2
DBUF3	DBUF4	DBUF5
DBUF6	MAPPED-BUFFER	MBUF1
MBUF2	MBUF3	MBUF4
MBUF5	MBUF6	RETURNED-BUFFER
SAVCTP	SAVREQ	SAVRSP
SAVSCS	SCSCMD	SCSEVT
SCSRSP	SCSSAV	

*WAIT

The WAIT command used only for writing your own tests. It is used to wait for one of the following:

CONNECTION-STATUS STATE, one of the following:

ACCEPT-REQUEST-SENT	CLOSED
CONNECT-REQUEST-RECEIVED	CONNECT-REQUEST-SENT
CONNECT-RESPONSE-RECEIVED	CONNECTION-IS-OPEN
DISCONNECT-REQUEST-RECEIVED	DISCONNECT-REQUEST-SENT
DISCONNECT-RESPONSE-RECEIVED	LISTENING
REJECT-REQUEST-SENT	WAITING-FOR-DISCONNECT-RESPONSE

CONNECTION-STATUS FLAG, of one of the following:

DATAGRAM-AVAILABLE-FLAG	DMA-TRANSFER-COMPLETE
EVENT-PENDING-FLAG	MESSAGE-AVAILABLE-FLAG

EVENT-STATUS, one of the following:

CONNECT-TO-LISTEN	CONNECTION-WAS-ACCEPTED
CONNECTION-WAS-REJECTED	CREDIT-IS-AVAILABLE
LITTLE-CREDIT-LEFT	MAINT-DATA-XFER-COMPLETE
MESSAGE-DATAGRAM-SEND-COMPLETE	NODE-CAME-ONLINE
NODE-WENT-OFFLINE	OK-TO-SEND-DATA
PORT-BROKE-CONNECTION	REMOTE-INITIATED-DISCONNECT
VC-BROKEN	

!All scripts here
*DEFAULT

SEQ 0120

The DEFAULT script runs the same tests as the ALL-TESTS script except it selects all nodes available for testing first. Type HELP ALL-TESTS for a listing of the tests run by that script.

*ALL-TESTS

The ALL-TESTS script runs of the following tests:

- TST01 Connect/Disconnect Test
- TST02 Function Set Request/Response Test
- TST10 Basic Message Test
- TST11 Message Test Count 4, Four Data Types
- TST12 Message Test Repeat 3, Four Data Types
- TST13 Message Test Delay 4, Four Data Types
- TST14 Message Test Size 1 to Maximum, Four Data Types
- TST30 Basic Datagram Test
- TST31 Datagram Test Size 4, Four Data Types
- TST32 Datagram Test Repeat 3, Four Data Types
- TST33 Repeat and Delay 4, Four Data Types
- TST40 Datagram Test Under Message Service
- TST50 Buffer Map and Buffer Unmap Test
- TST51 Buffer Map, Controller Read, Buffer Unmap
- TST52 Buffer Map, Buffer Move, Buffer Unmap
- TST53 Buffer Move (Delay 4)
- TST54 Buffer Move (Repeat 3)
- TST55 Buffer Move (Variable Length)
- TST56 Buffer Move (Size 1)
- TST57 Buffer Move (Multiple 1)
- TST58 Buffer Move (Exerciser-Responder Write) HSC50
- TST59 Buffer Move (Exerciser-Responder Write) TOPS-10/20/VMS
- TST80 Read Counter Test

*BASIC-TESTS

The BASIC-TESTS script runs the following tests:

- TST01 Connect/Disconnect Test
- TST02 Function Set Request/Response Test

*BUFFER-TESTS

The BUFFER-TESTS script runs the following tests:

- TST50 Buffer Map and Buffer Unmap Test
- TST51 Buffer Map, Controller Read, Buffer Unmap
- TST52 Buffer Map, Buffer Move, Buffer Unmap
- TST53 Buffer Move (Delay 4)
- TST54 Buffer Move (Repeat 3)
- TST55 Buffer Move (Variable Length)
- TST56 Buffer Move (Size 1)
- TST57 Buffer Move (Multiple 1)
- TST58 Buffer Move (Exerciser-Responder Write) HSC50
- TST59 Buffer Move (Exerciser-Responder Write) TOPS-10/20/VMS

*COUNTER-TEST

The COUNTER-TEST script runs the following test:

*DATA-TESTS

The DATA-TESTS script runs the following tests:

- TST30 Basic Datagram Test
- TST31 Datagram Test Size 4, Four Data Types
- TST32 Datagram Test Repeat 3, Four Data Types
- TST33 Repeat and Delay 4, Four Data Types
- TST40 Datagram Test Under Message Service

*MESSAGE-TESTS

The MESSAGE-TESTS script runs the following tests:

- TST10 Basic Message Test
- TST11 Message Test Count 4, Four Data Types
- TST12 Message Test Repeat 3, Four Data Types
- TST13 Message Test Delay 4, Four Data Types
- TST14 Message Test Size 1 to Maximum, Four Data Types

!All tests here

*TST01

TST01 Connect/Disconnect Test: This test will connect to the selected node and verify proper status (Event Pending, Connection Accepted, and Connection Open). The test will then disconnect from the selected node and verify proper status (Connection Closed).

*TST02

TST02 Function Set Request/Response: This test will connect to the selected node and verify proper status. A Function Set Request will be sent and proper status verified. The Response message is read and verified. The test will then disconnect from the selected node and verify proper status.

*TST10

TST10 Basic Message Test: This test will connect to the selected node and verify proper status. A Generate Message Request with a word count of 0 will be sent and proper status verified. The Response message is read and verified. The test will then disconnect from the selected node and verify proper status.

*TST11

TST11 Message Test Count 4, Four Data Types: For each data type, a Generate Message Request with a word count of 4 and buffer fill data of one of the data types will be sent and proper status verified. The Response message is read and verified. The data field is verified.

*TST12

TST12 Message Test Repeat 3, Four Data Types: For each of the data types, a Generate Message request with a word count of 4 and buffer fill data of the data type and a repeat count of 3 (which will generate 4 response messages) will be sent and proper status verified.

Each response message is read and verified. The data field is verified.

SEQ 0122

*TST13

TST13 Message Test Delay 4, Four Data Types: For each data type A Generate Message request with a word count of 4 and buffer fill data and a delay count of 4 (2 seconds) will be sent and proper status verified. The response message is read and verified The data field is verified as 4 bytes of 375.

*TST14

TST14 Message Test Size 1 to Maximum, Four Data Types: For each data type, a Generate Message Request with a variable byte count of 1 to maximum supported in the responder and buffer fill data will be sent and proper status verified. The Response messages are read and verified. The data field is verified as the correct number of bytes of fill data.

*TST30

TST30 Basic Datagram Test: This test will connect to the selected node and verify proper status. A Generate Datagram Request with a word count of 0 will be sent and proper status verified. The Response Datagram is read and verified. The test will then disconnect from the selected node and verify proper status.

*TST31

TST31 Datagram Test Size 4, Four Data Types: For each data type, a Generate Datagram Request with a word count of 4 and buffer fill data will be sent and proper status verified. The Response Datagram is read and verified. The data field is verified.

*TST32

TST32 Datagram Test Repeat 3, Four Data Types: For each data type, a Generate Datagram Request with a byte count of 4 and buffer fill data and a repeat count of 3 (which will generate 4 response datagrams) will be sent and proper status verified. Each Response Datagram is read and verified. The data field is verified.

*TST33

TST33 Datagram Test Repeat and Delay 4, Four Data Types: For each data type, a Generate Datagram Request with a word count of 4 and buffer fill data and a delay count of 4 (2 seconds) will be sent and proper status verified. The Response Datagram is read and verified. The data field is verified as 4 bytes of 375.

*TST40

TST40 Datagram Test Under Message Service: A Generate Datagram Request under Message Service with a word count of 26, buffer fill data of 201, a delay count of 4 (2 seconds), and a repeat count of 1 will be sent and proper status verified. The Response Datagram is read and verified. The data field is verified as 4 bytes of 201. The ending response message is read and verified The data field is verified as 4 bytes of 201.

*TST50

TST50 Buffer Map and Buffer Unmap Test: A Buffer Map Request for a real buffer with a buffer fill of 377 and a buffer length of 576 bytes is sent and proper status verified. The response message is read and verified. A Buffer Unmap Request for the same buffer is sent and proper status is verified. The Response Message is read and verified.

*TST51

TST51 Buffer Map, Controller Read, Buffer Unmap: For each data type, a Buffer Map Request for a real buffer with a buffer fill and a buffer length of 4 bytes is sent and proper status verified. The Response Message is read and verified. An internal buffer is mapped (in the exerciser), and bytes of 125 are written into the buffer. Data is read, and the buffer is verified to contain bytes of fill data. The internal buffer is unmapped. A Buffer Unmap Request for the responder buffer is sent and proper status is verified. The Response Message is read and verified.

*TST52

TST52 Buffer Map, Buffer Move, Buffer Unmap: A Buffer Map Request for a real buffer with a buffer fill of 252 and a buffer length of 4 bytes is sent and proper status verified. The Response Message is read and verified. An internal buffer is mapped in the exerciser and bytes of 125 are written into the buffer. A Move Buffer Command is issued for a responder write of 4 bytes with a delay, a repeat count, a packet multiplier and a packet size of 0 is sent and proper status verified. The Response Message is read and verified. The buffer is verified to contain bytes of 252. The internal buffer is unmapped. A Buffer Unmap Request for the same buffer is sent and proper status is verified, and the Response Message is read and verified.

*TST53

TST53 Buffer Move (Delay 4): This test is identical to TST53 except buffer fill data of 371 and delay count of 4 (2 seconds) is used.

*TST54

TST54 Buffer Move (Repeat 3): This test is identical to TST53 except buffer fill data of 370 and repeat count of 3 (generating 4 buffer moves) is used.

*TST55

TST55 Buffer Move (Variable Length): The following is performed starting with a buffer length of 4 and incrementing the buffer length by four until a buffer length of 1024 is reached. A buffer map request for a real buffer with a buffer fill of byte pair and a buffer length of 1024 bytes is sent and proper status verified. The Response Message is read and verified. Map an internal buffer 1024 bytes in length fill it with bytes of 125. Issue a Move Buffer Command for a responder write of variable length bytes with a delay, a repeat count, a packet multiplier and a packet size of 0. Proper status is verified. The Response Message is read and verified. Verify the buffer contains byte pair data. A Buffer Unmap request for the same buffer is sent and proper status is verified. The response message is read and

verified. The internal buffer is unmapped.

SEQ 0124

*TST56

TST56 Buffer Move (Size 1): A buffer map request for a real buffer with a buffer fill of 366, a buffer length of 1024 bytes and a packet size of 1 is sent and proper status verified. The response message is read and verified. An internal buffer is mapped and filled with bytes of 125. A Move Buffer command for a responder write of 1024 bytes with a delay, a repeat count, a packet multiplier of 0 and a packet size of 1 is sent and proper status verified. The Response Message is read and verified. The buffer is verified to contain 366. A Buffer Unmap request for the same buffer is sent and proper status is verified. The response message is read and verified. The internal buffer is unmapped.

*TST57

TST57 Buffer Move (Multiple 1): This test is identical to TST56 except a packet multiple of 1 is used.

*TST58

TST58 Buffer Move (Exerciser-Responder Write): This test runs only against the HSC responder. A buffer map request for a real buffer with a buffer fill of 252 and a buffer length of 1024 bytes is sent and proper status verified. The Response Message is read and verified. Map first internal buffer in the exerciser and fill with bytes of 125. A Move Buffer Request for an exerciser write of 1024 bytes with a delay, a repeat count, a packet multiplier and a packet size of 0 is sent. Proper status is verified. The Response Message is read and verified. The first internal buffer is unmapped. Map a second internal buffer and fill with 0 bytes. A Move Buffer Request for a responder write of 1024 bytes with a delay, a repeat count, a packet multiplier and a packet size of 0 is issued. Proper status is verified. The Response Message is read and verified. The buffer is verified to contain bytes of 125. A Buffer Unmap Request for the same buffer is sent and proper status is verified. The response message is read and verified. Unmap the second internal buffer.

*TST59

TST59 Buffer Move (Exerciser-Responder Write): This test will not run against the HSC responder. Map a buffer in the responder, a Buffer Map Request for a real buffer with a buffer fill of 252 and a buffer length of 1024 bytes is sent and proper status verified. The Response Message is read and verified. Map an internal buffer of 1024 bytes and fill with bytes of 125. Transfer the data to the responder from the exerciser. Test for data transfer complete. Unmap the first internal buffer. Map a second internal buffer fill with 0 bytes. Issue a Move Buffer Request for a responder write of 1024 bytes with a delay, a repeat count, a packet multiplier and a packet size of 0 is sent and proper status verified. The response message is read and verified. Verify the buffer contains bytes of 125. A Buffer Unmap Request is sent and proper status is verified. The Response Message is read and verified.

*TST80

TST80 Read Counter Test: This test will connect to the selected node

and verify proper status. A Read Counter Request is issued and proper status is verified. The Response Message is read and verified. The number of acks, nacks and no responses for path A and path B is printed out along with the number of disregarded datagrams.

*TST99

TST99 CTP Exerciser: This test runs 3 iterations of the exerciser sequence while varying the repeat counts to 2, 32, and 64. This is done to increase the amount of traffic on the CI. The exerciser sequence utilizes the three basic methods of transfers (1) messages, (2) datagrams, and (3) buffer transfers, to create traffic on the CI.

The test begins by initializing the test tables and flags and goes on to get configuration data on all nodes on the CI. It uses this configuration data and the selected nodes to create selection list for the exerciser. Nodes that are selected but not accessible are dropped.

The exerciser sequence consists of the following for each node. If no /ITERATION switch is specified in the RUN command, then 100 iterations of the exerciser are performed. All operations are performed on each node before moving to the next item in the list:

- o Connect to node.
- o Issue Generate Message Commands of 64 bytes.
- o Wait for and verify the Generate Message Responses.
- o Issue Generate Datagram Requests of 20 bytes.
- o Wait for and verify Generate Datagram Responses.
- o Map internal buffer, Issue Map Buffer Request for 64 bytes, verify Map Buffer Response.
- o Issue Move Buffer Request, wait for and verify Move Buffer Response.
- o Verify move buffer data transfers.
- o Issue Unmap Buffer Request, verify Unmap Buffer Response, Unmap internal buffer.
- o Disconnect from node.

!End of DFCIB.HLP

```

1      ;Z:<GSCOTT.DFCIB>DFCIBT.MAC.32 21-Aug-85 12:37:54, Edit by GSCOTT
2      ;CMDERR macro should go to top level parser rather than returning inline
3      ;Z:<GSCOTT.DFCIB>DFCIBT.MAC.30 21-Aug-85 11:55:48, Edit by GSCOTT
4      ;Change major version to 2.
5      ;Z:<GSCOTT.DFCIB>DFCIBT.MAC.24 14-Aug-85 17:03:09, Edit by GSCOTT
6      ;Default ERROR1 parameters correctly if they are missing.
7      ;Z:<GSCOTT.DFCIB>DFCIBT.MAC.17 7-Aug-85 13:00:47, Edit by GSCOTT
8      ;Remove getd/putd
9      ;Z:<GSCOTT.DFCIB>DFCIBT.MAC.14 7-Aug-85 00:51:11, Edit by GSCOTT
10     ;Remove SO. Add ERROR1 macro
11     ;Z:<GSCOTT.DFCIB>DFCIBT.MAC.11 6-Aug-85 14:36:42, Edit by GSCOTT
12     ;Add SO=.A14 temporarily
13     ;Z:<GSCOTT.DFCIB>DFCIBT.MAC.9 6-Aug-85 11:07:54, Edit by GSCOTT
14     ;Define GLXLIB symbols that aren't in the GLXMAC on the -20
15     ;Z:<GSCOTT.DFCIB>DFCIBT.MAC.8 6-Aug-85 11:04:03, Edit by GSCOTT
16     ;Add CMDERR and GUIDE macros.
17     ;Z:<GSCOTT.DFCIB>DFCIBT.MAC.3 5-Aug-85 11:40:38, Edit by GSCOTT
18     ;Start conversion to GLXLIB
19
20     000002      DECVER==2      ;Major version
21     000020      VEDIT==20     ;Edit number: never reset to 0
22
23     DEFINE NAME (DECVER,VEDIT),<
24     UNIVER DFCIBT CI20 Exerciser Symbols Version DECVER('VEDIT')>
25
26     NAME \DECVER,\VEDIT^
27
28     ;Set version
29
30     000137      LOC 137      ;Get to absolute location .JBVER
31     000137, G0J200 000020    <DECVER>B11+VEDIT ;Set the major version and edit
32     000000      RELOC      ;Back to relocatable code
33
34     ;Assembly parameters
35
36     NOSYM
37     SEARCH GLXMAC,MONSYM,UUOSYM
38     SALL
39
40     .TEXT  '',/RUNAME:DFCIB' ;set EXE file name
  
```

```

41          SUBTTL  Macros
42
43      ;Define NAME macro for other modules
44
45      DEFINE NAME      (MODULE,DECVER,VEDIT,TEXT),<
46
47          TITLE  MODULE TEXT version DECVER('VEDIT')>
48
49      ;Define command error macro
50
51      DEFINE  CMDERR  (TEXT),<
52          JRST [ $CALL  CMDLOG
53                  $TEXT  (,<? TEXT>)
54                  JRST  TOPLVL]>
55
56      ;Define macro to output guide words
57
58      DEFINE  GUIDE  (TEXT),<
59          MOVEI  S2,[FLDDB. (.CMNOI,,<Point 7,[ASCIZ/TEXT/]>)]
60          $CALL  CMDPRS
61      >
62
63      ;Define ERROR1 macro like KCSUB uses
64      ;ERROR1(flag,correct,actual,function-testing,comment,routine-to-call)
65
66      000013          EM=.A13          ;Error message pointer
67
68      DEFINE  ERROR1  (X,CORR,ACTU,FUNCT,CMNT,ROUT),<
69          $CALL  [JSP  EM,ERRHAN
70                  IFNB  <X>,<X>
71                  IFB   <X>,<0>
72                  IFNB  <CORR>,<CORR>
73                  IFB   <CORR>,<0>
74                  IFNB  <ACTU>,<ACTU>
75                  IFB   <ACTU>,<0>
76                  IFNB  <FUNCT>,<[ASCIZ\FUNCT\]>
77                  IFB   <FUNCT>,<0>
78                  IFNB  <CMNT>,<[ASCIZ\CMNT\]>
79                  IFB   <CMNT>,<0>
80                  IFNB  <ROUT>,<ROUT>
81                  IFB   <ROUT>,<0>
82      ]>;End of DEFINE ERROR1
83

```

```
84          SUBTTL  GLXLIB Symbols
85
86          ;Define symbols that aren't in the GLXMAC on the -20
87
88          000001      .FDSTR==.FDFIL      ;Structure containing the file
89          000002      .FDNAM==.FDFIL+1    ;The file name
90          000003      .FDEXT==.FDFIL+2    ;The extension
91          000004      .FDPPN==.FDFIL+3    ;The PPN
92          000005      .FDPAT==.FDFIL+4    ;Start of the path
93
94
95          END
```

NO ERRORS DETECTED

PROGRAM BREAK IS 000000
CPU TIME USED 00:01.456

127P CORE USED

DECVER	20#	26	31		
EM	67#				
VEDIT	21#	26	31		
.A13	67				
.FDEXT	90#				
.FDFIL	88	89	90	91	92
.FDNAM	89#				
.FDPAT	92#				
.FDPPN	91#				
.FDSTR	88#				

SEQ 0129

CMDERR
ERROR1
GUIDE
NAME

51#
69#
58#
23#

26

45#

SEQ 0130

DFCIBI - Bootstrap code for GALAXY runtime system for DCIB MACRO %53B(1242) 17:32 27-Aug-85 Page 1
DFCIBI MAC 7-Aug-85 12:27 Essentially GLXINI for -10 and -20 both /GAS

SEQ 0131

;Z:<GSCOTT.DFCIB>DFCIBI.MAC.2 7-Aug-85 12:26:43, Edit by GSCOTT

TITLE DFCIBI - Bootstrap code for GALAXY runtime system for DCIB
SUBTTL Essentially GLXINI for -10 and -20 both /GAS

SEARCH GLXMAC,MONSYM,MACSYM,UUOSYM
PROLOG(GLXINI,INI)^

^

U00021
000025

INIMAN==:21 ;Maintenance edit number
INIDEV==:25 ;Development edit number
VERGIN (INI) ;Generate edit number

;Note: Depends on MONTYP being set before call to I%INIT
; MONTYP/ -1 = TOPS-20
; MONTYP/ 0 = TOPS-10

EXTERN MONTYP

```

19      SUBTTL User referenced entry vector
20
21
22      ; All references to the OTS by the user are through ENTVEC table
23      ;
24      000000
25      DEFINE ZZ==0                                ;; INITIALIZE COUNTER
26      CDO (X), <                                ;; MACRO TO DEFINE ENTRY POINTS
27      IFNB <X>, <X::>                            ;; DEFINE GLOBAL IF NOT NULL
28      JRST @.+1                                    ;; DISPATCH
29      EXP Z.                                       ;; STORE INDEX INTO GLXOTS
30      ZZ==ZZ+1                                    ;; POINT TO NEXT ENTRY
31      >                                           ;; END OF CDO MACRO
32
33      000000' ENTVEC::LIBVEC                        ;DEFINE ALL ENTRY POINTS
34      000610' 777777 777777 EXP -1                ;TERMINATE VECTOR
35
36      000611' WORK: BLOCK 20                        ;Get space for working
37      000631' ARG: BLOCK 6                         ;Get space for argument block...
38                                           ;Note: this space can also be used as
39                                           ;spillover from WORK
40
41      ; Set up correct places to search for the library depending on the system.
42      000637' 44 63 53 00 00 00 DSKD10: SIXBIT/DSK/
43      000640' 63 71 63 00 00 00 SYSD10: SIXBIT/SYS/
44
45      000641' 000000 001036' DSKD20: [ASCIZ/DSK:/]
46      000642' 000000 001037' SYSD20: [ASCIZ/SYS:/]
    
```

```

47      SUBTTL I%INIT - Initialize the runtime system
48
49
50      ;Entry point for segment and module initialization
51
52      ;CALL IS:      S1/ Length of IB (Runtime Initialization Block)
53      ;              S2/ Address of IB
54
55      ENTRY  I%INIT
56
57      I%INIT: PUSH  P,S1          ;SAVE INPUT ARGS AWAY
58              PUSH  P,S2          ;
59              PUSHJ P,CHKLIB      ;SEE IF THE LIBRARY IS ALREADY THERE
60              JUMPT INIT.2        ;IT IS - SKIP LOTS OF STUFF
61              PUSHJ P,ACQLIB      ;ACQUIRE THE LIBRARY SYSTEM
62              SKIPE ENTVEC+1      ;Is the entry vector initialized?
63              JRST  INIT.2        ;Yes, don't do it again.
64              SETZM S2            ;Set up index
65
66      INIT.1: ADDM  S1,ENTVEC+1(S2) ;ADJUST INDEX INTO GLXOTS
67              ADDI  S2,2          ;ACCOUNT FOR TWO WORD ENTRIES
68              SKIPL ENTVEC(S2)    ;END OF VECTOR ?
69              JRST  INIT.1        ;NO - LOOP
70
71      INIT.2: POP   P,S2          ;RESTORE USER'S S2
72              POP   P,S1          ;RESTORE USER'S S1
73
74      BEGIN:: HRLI  S1,%GLX      ;LOAD VERSION
75              PUSHJ P,ENTVEC      ;CONTINUE INITIALIZATION
76              $RETT              ;RETURN TO CALLER
    
```

```

77      SUBTTL ACQLIB - Acquire the run-time system
78
79
80      ; Here if the OTS is not in place. If this is a debugging world,
81      ; look first in DSK:. If that fails, or if we are not debugging,
82      ; look in SYS:. If not found, give an error message and quit.
83      ; If found, return the address of the dispatch table for the library
84      ; in S1.
85
86      ; Call:          No arguments
87
88      ; Return:        S1 / Address of OTS dispatch table (returned from TRYSEG)
89
90      000664' 336 00 0 00 000135 ACQLIB: SKIPN DEBUGW ;Debugging ?
91      000665' 254 00 0 00 000673' JRST ACQL.1 ;Nope - skip checking on DSK
92      000666' 332 00 0 00 000000* SKIPE MONTYP ;Skip if TOPS-10
93      000667' 334 01 0 00 000641' SKIPA S1,DSKD20 ;TOPS-20
94      000670' 200 01 0 00 000637' MOVE S1,DSKD10 ;Try generic DSK on -10
95      000671' 260 17 0 00 000733' PUSHJ P,TRYSEG ;Try it again
96      000672' 326 00 0 00 000712' JUMPT ACQL.2 ;If sucessful - finish up
97
98      000673' 332 00 0 00 000666* ACQL.1: SKIPE MONTYP ;Skip if TOPS-10
99      000674' 334 01 0 00 000642' SKIPA S1,SYSD20 ;TOPS-20
100     000675' 200 01 0 00 000640' MOVE S1,SYSD10 ;Try SYS
101     000676' 260 17 0 00 000733' PUSHJ P,TRYSEG ;One more time
102     000677' 326 00 0 00 000712' JUMPT ACQL.2 ;If sucessful - finish up
103
104     000700' 336 00 0 00 000673* SKIPN MONTYP ;Skip if -20
105     OUTSTR [ASCIZ :? GLXINI - Unable to obtain run-time system
106     ;]
107     000701' 051 03 0 00 001040' SKIPN MONTYP ;Skip if -20
108     000702' 352 00 0 00 000700* MONRT. ;Stop
109     000703' 047 01 0 00 000012' HRROI S1,[ASCIZ \? GLXINI - Unable to obtain run-time system
110     ;\]
111     000704' 561 01 0 00 001040' SKIPE MONTYP ;Skip if -10
112     000705' 332 00 0 00 000702* PSOUT
113     000706' 104 00 0 00 000076' SKIPE MONTYP ;Skip if -10
114     000707' 332 00 0 00 000705* HALTF
115     000710' 104 00 0 00 000170' JRST ACQL.1 ;We can continue
116     000711' 254 00 0 00 000673'
117     000712' 263 17 0 00 000000 ACQL.2: POPJ P, ;Return
  
```


SUBTTL CHKLIB - Routine to test if library is in place

;CHKLIB looks for the first page of the OTS library. If this page
 ; is there, it is assumed that the library was loaded with the
 ; calling program, or has been pre-merged.

;CALL IS: No arguments

;TRUE RETURN: Library is in place

;FALSE RETURN: Library is not there

CHKL1B:	SKIPE	MONTYP	;Skip if TOPS-10
	JRST	CHKL20	;TOPS-20
	HRRZ	S1,JBHRL	;Get highest addr in high seg
	SKIPN	S1	;Is there a segment? Yes, skip
	TDZA	TF,TF	;LOAD 'FALSE' INTO TF
	MOVX	TF,TRUE	;ELSE ITS THERE
	POPJ	P,	;RETURN IN EITHER CASE
CHKL20:	MOVE	S1,JBHSL	;FIRST PAGE OF LIBRARY
	JUMPE	S1,[MOVX TF,FALSE	;If zero, then not set up
		JRST CHKL.1]	;Return false
	HRLI	S1,FHSLF	;FOR THIS PROCESS,
	RPACS		;READ THE ACCESS BITS
	ERJMP	.+2	;IF FAILS, SAY OTS NOT THERE
	TXNN	S2,PAZPEX	;CHECK FOR PAGE'S EXISTENCE
	TDZA	TF,TF	;IF PAGES DON'T EXIST, RETURN
	MOVX	TF,TRUE	;ELSE TRUE,INDICATING OTS IN PLACE
CHKL.1:	POPJ	P,	;RETURN

118
 119
 120
 121
 122
 123
 124
 125
 126
 127
 128
 129
 130 000713' 332 00 0 00 000707*
 131 000714' 254 00 0 00 000722'
 132 000715' 550 01 0 00 000000*
 133 000716' 336 00 0 00 000001
 134 000717' 634 00 0 00 000000
 135 000720' 561 00 0 00 777777
 136 000721' 263 17 0 00 000000
 137
 138 000722' 200 01 0 00 000075
 139
 140 000723' 322 01 0 00 001052'
 141 000724' 505 01 0 00 400000
 142 000725' 104 00 0 00 000057
 143 000726' 320 16 0 00 000730'
 144 000727' 607 02 0 00 010000
 145 000730' 634 00 0 00 000000
 146 000731' 561 00 0 00 777777
 147 000732' 263 17 0 00 000000

```

148 SUBTTL TRYSEG - Routine to fetch runtime system
149
150 ;CALL IS: S1/ Device to get runtime system from in SIXBIT (TOPS10)
151 ; Address of device string (TOPS20)
152
153 ;TRUE RETURN: Segment is in place
154 ; with S1/ Address of dispatch table
155 ;FALSE RETURN: Segment could not be obtained
156
157 002000 JS.X0==2000 ;EXECUTE-ONLY BIT IN JOB STATUS WORD
158
159 000733' 332 00 0 00 000713* TRYSEG: SKIPE MONTYP ;Skip if TOPS-10
160 000734' 254 00 0 00 000767' JRST TRYS20 ;TOPS-20
161
162 000735' 202 01 0 00 000631' MOVEM S1,ARG+0 ;STORE DEVICE NAME
163 000736' 201 01 0 00 000611' MOVEI S1,WORK+0 ;BLT the acs away
164 000737' 251 01 0 00 000630' BLT S1,WORK+17 ;STORE THEM
165 000740' 200 01 0 00 001054' MOVE S1,['GLXLIB'] ;Name of OTS
166 000741' 202 01 0 00 000632' MOVEM S1,ARG+1 ;STORE THE FILE NAME
167 000742' 561 02 0 00 000000 HRROI S2,..GTSTS ;GET JOB STATUS
168 000743' 047 02 0 00 000041 GETTAB S2, ;FROM THE MONITOR
169 000744' 400 02 0 00 000000 SETZ S2, ;CAN'T - ASSUME NOT EXECUTE ONLY
170 000745' 606 02 0 00 002000 TRNN S2,JS.X0 ;EXECUTE ONLY?
171 000746' 634 02 0 00 000002 TDZA S2,S2 ;NO
172 000747' 201 02 0 00 200000 MOVX S2,UU.PHY ;YES - SET FOR PHYSICAL ONLY GETSEG
173 000750' 670 02 0 00 001055' TXO S2,<GETSEG S1,> ;FORM INSTRUCTION
174 000751' 201 01 0 00 000631' MOVEI S1,ARG ;GET LOCATION OF BLOCK
175 000752' 256 00 0 00 000002 XCT S2, ;ATTACH HIGH SEGMENT
176 000753' 634 00 0 00 000000 TDZA TF,TF ;IF FAILS, FAKE A FALSE
177 000754' 561 00 0 00 777777 MOVX TF,TRUE ;ELSE LOAD A 'TRUE'
178 000755' 201 01 0 00 001056' MOVE S1,[WORK+S2,..S2] ;RESTORE SOME ACS
179 000756' 251 01 0 00 000017 BLT S1,17 ;
180 000757' 336 00 0 00 000000 SKIPN TF, ;True?
181 000760' 263 17 0 00 000000 POPJ P, ;Return
182
183 000761' 200 01 0 00 001057' MOVE S1,[XWD -2,..GTUPM] ;Get UPM word for our high seg.
184 000762' 047 01 0 00 000041 GETTAB S1, ;Ask the monitor
185 000763' 505 01 0 00 000000* HRLI S1,..JBHGH ;Monitor won't give it, assume 400000
186 000764' 557 00 0 00 000001 HLRZS S1, ;Get high seg start adr
187 000765' 271 01 0 00 000000* ADDI S1,..JBHDA ;Point past JOB DAT
188
189 000766' 263 17 0 00 000000 POPJ P, ;RETURN
  
```

190											
191	000767'	200	00	0	00	000001	TRY20:	MOVE	TF,S1		;Save device string a second
192	000770'	201	01	0	00	400000		MOVX	S1,.,FHSLF		;For self
193	000771'	104	00	0	00	000205		GEVEC			;Get entry vector word
194	000772'	261	17	0	00	000002		PUSH	P,S2		;And save it
195	000773'	200	01	0	00	000000		MOVE	S1,TF		;Restore s1
196											
197	000774'	505	01	0	00	440700		HRLI	S1,(POINT 7,0)		;Make the address a byte pointer
198	000775'	200	02	0	00	001060'		MOVE	S2,[POINT 7,WORK]		;Generate a pointer to destination for
199											; string
200	000776'	134	00	0	00	000001	TRY2.1:	ILDB	TF,S1		;Get a byte of the device string
201	000777'	322	00	0	00	001002'		JUMPE	TF,TRY2.2		;Done?
202	001000'	136	00	0	00	000002		IDPB	TF,S2		;No, save the byte
203	001001'	254	00	0	00	000776'		JRST	TRY2.1		;Go get another
204											
205	001002'	200	01	0	00	001064'	TRY2.2:	MOVE	S1,[POINT 7,OTSNAM]		;Get the name of the OTS
206											
207	001003'	134	00	0	00	000001	TRY2.3:	ILDB	TF,S1		;Get a byte of the OTS name
208	001004'	136	00	0	00	000002		IDPB	TF,S2		;Save the byte
209	001005'	326	00	0	00	001003'		JUMPN	TF,TRY2.3		;Get till done
210											
211	001006'	205	01	0	00	100001		MOVX	S1,GJ%SH+GJ%OLD		;SHORT GTJFN, OLD FILE
212	001007'	200	02	0	00	001060'		MOVE	S2,[POINT 7,WORK]		;Get the pointer to the beginning of
213											; the string
214	001010'	104	00	0	00	000020		GTJFN			;TRY TO GET A HANDLE ON THE FILE
215	001011'	254	00	0	00	001033'		JRST	TRY2.4		;FAKE A FALSE RETURN
216	001012'	660	01	0	00	200000		TXO	S1,GT%ADR		;SET BIT FOR ADDRESS SELECTION
217	001013'	505	01	0	00	400000		HRLI	S1,.,FHSLF		;FOR MYSELF
218	001014'	200	02	0	00	001065'		MOVE	S2,[1,.,760]		;ALL PAGES FROM 1 TO 760
219	001015'	104	00	0	00	000200		GET			;GET THE SEGMENT
220	001016'	320	16	0	00	001033'		ERJMP	TRY2.4		;IF GET FAILS, RETURN FALSE
221											
222	001017'	201	01	0	00	400000		MOVX	S1,.,FHSLF		;For self
223	001020'	104	00	0	00	000205		GEVEC			;Get entry vector word for OTS
224	001021'	553	02	0	00	000002		HRRZS	S2,S2		;Get only the address
225	001022'	200	00	0	00	000002		MOVE	TF,S2		;Save it
226	001023'	242	02	0	00	777767		ADR2PG	S2		;Make a page number
227	001024'	202	02	0	00	000075		MOVEM	S2,.,JBHSD		;Save it for DDT and CHKLIB
228	001025'	522	02	0	00	000715*		HLLDM	S2,.,JBHRL##		;Convince DDT to use the symbols
229	001026'	262	17	0	00	000002		POP	P,S2		;Restore entry vector word
230	001027'	104	00	0	00	000204		SEVEC			;Set it back
231	001030'	200	01	0	00	000000		MOVE	S1,TF		;Get address of dispatch vector
232	001031'	561	00	0	00	777777		MOVX	TF,TRUE		;We are ok
233	001032'	254	00	0	00	001035'		JRST	TRY2.5		;go to return
234											
235	001033'	201	00	0	00	000000	TRY2.4:	MOVX	TF,FALSE		;HERE FOR FALSE RETURN
236	001034'	262	17	0	00	000002		POP	P,S2		;Restore stack
237	001035'	263	17	0	00	000000	TRY2.5:	POPJ	P,		;AND RETURN

DFCIBI - Bootstrap code for GALAXY runtime system for DCIB MACRO %53B(1242) 17:32 27-Aug-85 Page 8
DFCIBI MAC 7-Aug-85 12:27 TRYSEG - Routine to fetch runtime system

SEQ 0138

238
239
240 ;Literals
241
242 001036'
243 INI%L: END

NO ERRORS DETECTED

PROGRAM BREAK IS 001066
CPU TIME USED 00:03.817

147P CORE USED

ACQL.1	000673'	I%INIT	000643'	ent	M%IPRC	000244'	int	%%.OTS	515651	000000	spd
ACQL.2	000712'	I%INT1	000450'	int	M%IPRM	000246'	int	.AOS		000042'	int
ACQLIB	000664'	I%INT2	000452'	int	M%IPSN	000240'	int	.CPUTY		000070'	int
ARG	000631'	I%INT3	000454'	int	M%NXPG	000242'	int	.FHSLF		400000	sin
C%BRCV	000112' int	I%IOFF	000444'	int	M%RELP	000236'	int	.GTSTS		000000	spd
C%CPID	000116' int	I%ION	000442'	int	M%RLNP	000252'	int	.GTUPM		000100	spd
C%INTR	000104' int	I%JINF	000474'	int	M%RMEM	000262'	int	.JBHDA		000765'	ext
C%KPID	000120' int	I%NOW	000436'	int	M%RPAG	000302'	int	.JBHGH		000763'	ext
C%MAXP	000124' int	I%SLP	000446'	int	MONRT.	047040	000012'	.JBHRL		001025'	ext
C%PIDJ	000126' int	I%SOPR	000464'	int	MONTYP		000733'	.JBHSO		000075'	sin
C%RECV	000110' int	I%TIMR	000456'	int	OTSINI	051140	000000'	.POPJ		000050'	int
C%REL	000114' int	I%WTO	000466'	int	OUTSTR		000000	.POPJ1		000062'	int
C%RPRM	000102' int	INI%L	001036'		P		000017'	.RETE		000040'	int
C%SEND	000106' int	INIDEV	000025'	sin	PAXPEX	010000	000000	.RETF		000036'	int
C%SPID	000122' int	INIEDT	000025'	sin	PJRST	324740	000000	.RETT		000034'	int
CHKL.1	000732'	INIMAN	000021'	sin	PSOUT	104000	000076'	.SAVE1		000012'	int
CHKL20	000722'	INIT.1	000653'		RPACS	104000	000057'	.SAVE2		000014'	int
CHKLIB	000713'	INIT.2	000657'		S%CMND		000512'	.SAVE3		000016'	int
DEBUGW	000135' spd	JS.XO	002000	spd	S%DATI		000506'	.SAVE4		000020'	int
DSKD10	000637'	JUMPF	000000	spd	S%ERR		000520'	.SAVE8		000054'	int
DSKD20	000641'	JUMPT	000000	spd	S%EXIT		000524'	.SAVET		000022'	int
ENTVEC	000000' int	K%BACK	000554'	int	S%INTR		000522'	.SC2UD		000066'	int
ERJMP	320700	K%BIN	000552'	int	S%NUMI		000532'	.SOS		000044'	int
F%AOPT	000146' int	K%BOUT	000550'	int	S%SCMP		000516'	.STKST		000056'	int
F%CHKP	000174' int	K%BUFF	000560'	int	S%SIXB		000510'	.STOP		000052'	int
F%DEL	000204' int	K%FLSH	000562'	int	S%TBAD		000526'	.SV13		000024'	int
F%DREL	000164' int	K%OPEN	000564'	int	S%TBDL		000530'	.SV14		000026'	int
F%FCHN	000206' int	K%RCOC	000534'	int	S%TBLK		000514'	.SV15		000030'	int
F%FD	000200' int	K%SOUT	000546'	int	S1		000001'	.SV16		000032'	int
F%IBUF	000156' int	K%STYP	000542'	int	S2		000002'	.TRSET		000060'	int
F%IBYT	000152' int	K%SUET	000540'	int	SEGIN	104000	000661'	.UD2SC		000064'	int
F%INFO	000176' int	K%TPOS	000556'	int	SEVEC		000204'	.ZCHNK		000010'	int
F%IOPN	000144' int	K%TXTI	000544'	int	SYSD10		000640'	.ZERO		000046'	int
F%OBUF	000160' int	K%WCOC	000536'	int	SYSD20		000642'	.ZPAGA		000004'	int
F%OBYT	000154' int	L%APOS	000412'	int	T%TEXT		000326'	.ZPAGN		000006'	int
F%OOPN	000150' int	L%CBFR	000366'	int	T%TTY		000330'				
F%POS	000172' int	L%CENT	000364'	int	TF		000000'				
F%REL	000162' int	L%CLST	000360'	int	TRUE	777777	777777'				
F%REN	000202' int	L%CURR	000404'	int	TRYS.1		000776'				
F%REW	000170' int	L%DENT	000370'	int	TRYS.2		001002'				
F%RREL	000166' int	L%DLST	000362'	int	TRYS.3		001003'				
FALSE	000000' spd	L%FIRS	000374'	int	TRYS.4		001033'				
FTJSYS	777777	L%LAST	000376'	int	TRYS.5		001035'				
FTUOS		L%NEXT	000372'	int	TRYS20		000767'				
GET	104000	L%PREM	000402'	int	TRYSEG		000733'				
GETSEG	047000	L%PREV	000400'	int	UU.PHY		200000'	spd			
GETTAB	047000	L%RENT	000406'	int	WORK		000611'				
GEVEC	104000	L%SIZE	000410'	int	ZZ		000304'	spd			
GJ%OLD	100000	LSTIN.	000000'	spd	\$RET	263740	000000'	spd			
GJ%SHI	000001	M%ACQP	000234'	int	\$RETF	324740	000036'	spd			
GT%ADR		M%AGNP	000250'	int	\$RETI	322000	000050'	spd			
GTJFN	104000	M%CLNC	000254'	int	\$RETT	326000	000050'	spd			
HALTF	104000	M%FPGS	000256'	int	\$RETT	324740	000034'	spd			
I%EXIT		M%GMEM	000260'	int	%%.GLX		000010'	spd			
I%HOST		M%GPAG	000300'	int	%%.MOD	475470	515651	spd			

ACQL.1	91	98#	115												
ACQL.2	96	102	117#												
ACQLIB	61	90#													
ARG	36#	162	166	174											
CHKL.1	140	147#													
CHKL20	131	138#													
CHKLIB	59	130#													
DEBUGW	90														
DSKD10	42#	94													
DSKD20	45#	93													
ENTVEC	32#	62	66	68	75										
FALSE	139	235													
GJ%OLD	211														
GJ%SHT	211														
GT%ADR	216														
I%INIT	55	57#													
INI%L	242#														
INIDEV	11#	12													
INIEDT	12#														
INIMAN	10#	12													
INIT.1	66#	69													
INIT.2	60	63	71#												
JS.XO	157#	170													
LSTIN.	8	32													
MONTYP	18#	92	98	104	107	111	113	130	159						
P	8	57	58	59	61	71	72	75	95	101	117	136	147	181	
PA%PEX	189	194	229	236	237										
S1	144														
	57	66	72	74	93	94	99	100	109	132	133	138	139	141	
	162	163	164	165	166	173	174	178	179	183	184	185	186	187	
S2	191	192	195	197	200	205	207	211	216	217	222	231			
	58	64	66	67	68	71	144	167	168	169	170	171	172	173	
	175	178	194	198	202	208	212	218	224	225	226	227	228	229	
SEGIN	236														
SYSD10	74#														
SYSD20	43#	100													
TF	46#	99													
	134	135	139	145	146	176	177	180	191	195	200	201	202	207	
	208	209	225	231	232	235									
TRUE	135	146	177	232											
TRYS.1	200#	203													
TRYS.2	201	205#													
TRYS.3	207#	209													
TRYS.4	215	220	235#												
TRYS.5	233	237#													
TRYS20	160	191#													
TRYSEG	95	101	159#												
UU.PHY	172														
WORK	35#	163	164	178	198	212									
ZZ	24#														
%%.GLX	74														
..MX1	135#	135	136	139#	139	146#	146	147	172#	172	173	177#	177	178	
	192#	192	193	211#	211	212	222#	222	223	232#	232	233	235#	235	

SEQ 0140

[illegible]


```

1      ;Z:<GSCOTT.DFCIB>DFCIB1.MAC.135 27-Aug-85 11:17:51, Edit by GSCOTT
2      ;Add SSCBF% error code
3      ;Z:<GSCOTT.DFCIB>DFCIB1.MAC.132 23-Aug-85 14:58:08, Edit by GSCOTT
4      ;Add more SCSUO error codes
5      ;Z:<GSCOTT.DFCIB>DFCIB1.MAC.129 22-Aug-85 23:00:56, Edit by GSCOTT
6      ;Add calls to RTCHK
7      ;Z:<GSCOTT.DFCIB>DFCIB1.MAC.126 22-Aug-85 20:14:19, Edit by GSCOTT
8      ;Fix other messages to be a little better and readable
9      ;Z:<GSCOTT.DFCIB>DFCIB1.MAC.115 22-Aug-85 19:04:32, Edit by GSCOTT
10     ;Fix up CALLN output to be more useful
11     ;Z:<GSCOTT.DFCIB>DFCIB1.MAC.101 21-Aug-85 15:14:44, Edit by GSCOTT
12     ;Do the required DIAG UO on the 10.
13     ;Z:<GSCOTT.DFCIB>DFCIB1.MAC.93 21-Aug-85 12:05:19, Edit by GSCOTT
14     ;I guess the temp code is now permanent. Drop the "temp" part.
15     ;Z:<GSCOTT.DFCIB>DFCIB1.MAC.91 20-Aug-85 17:59:55, Edit by GSCOTT
16     ;Add temp code for TOPS-10 debug operation
17     ;Z:<GSCOTT.DFCIB>DFCIB1.MAC.89 19-Aug-85 15:28:16, Edit by GSCOTT
18     ;Remember SCS$ called PC here rather than trying to figure it out later
19     ;Z:<GSCOTT.DFCIB>DFCIB1.MAC.86 19-Aug-85 14:43:05, Edit by GSCOTT
20     ;Stored SCSERC from wrong AC on the 20
21     ;Z:<GSCOTT.DFCIB>DFCIB1.MAC.84 16-Aug-85 13:20:42, Edit by GSCOTT
22     ;RCON didn't set up the length of the connect block
23     ;Z:<GSCOTT.DFCIB>DFCIB1.MAC.81 15-Aug-85 21:47:52, Edit by GSCOTT
24     ;Make COP, CVER, CREV == for no DDT output
25     ;Z:<GSCOTT.DFCIB>DFCIB1.MAC.80 15-Aug-85 13:07:16, Edit by GSCOTT
26     ;Intern SCSERC, the last error code from SCS$.
27     ;Z:<GSCOTT.DFCIB>DFCIB1.MAC.78 15-Aug-85 11:07:14, Edit by GSCOTT
28     ;CONN didn't reset the arg block length
29     ;Z:<GSCOTT.DFCIB>DFCIB1.MAC.76 15-Aug-85 10:44:46, Edit by GSCOTT
30     ;Install GETTIM to get system uptime on -10 or -20
31     ;Z:<GSCOTT.DFCIB>DFCIB1.MAC.69 15-Aug-85 09:35:15, Edit by GSCOTT
32     ;Add TOPS-10 SCS UO support the right way
33     ;Z:<GSCOTT.DFCIB>DFCIB1.MAC.68 14-Aug-85 11:46:07, Edit by GSCOTT
34     ;Reorder externs by module
35     ;Z:<GSCOTT.DFCIB>DFCIB1.MAC.65 14-Aug-85 11:27:59, Edit by GSCOTT
36     ;Add routine SCS$ to perform SCS function and print error code
37     ;Z:<GSCOTT.DFCIB>DFCIB1.MAC.57 7-Aug-85 15:57:29, Edit by GSCOTT
38     ;Use MAPBBA (page from GLXLIB) for address of MAPBB.
39     ;Z:<GSCOTT.DFCIB>DFCIB1.MAC.43 6-Aug-85 17:10:36, Edit by GSCOTT
40     ;Remove ACO usage, remove routines no longer used (WAITC, WAITE).
41     ;Z:<GSCOTT.DFCIB>DFCIB1.MAC.37 6-Aug-85 16:52:10, Edit by GSCOTT
42     ;Change ac names, use $TEXT, $SAVE, etc.
43     ;Z:<GSCOTT.DFCIB>DFCIB1.MAC.28 6-Aug-85 14:35:08, Edit by GSCOTT
44     ;WAITX time to S1
45     ;Z:<GSCOTT.DFCIB>DFCIB1.MAC.26 5-Aug-85 13:40:32, Edit by GSCOTT
46     ;GPRSB argument into S1 from ACO
47     ;Z:<GSCOTT.DFCIB>DFCIB1.MAC.23 5-Aug-85 13:35:37, Edit by GSCOTT
48     ;Begin GLXLIB conversion: RCON arguments in S1, S2.
49
50     ; 5-7-85 ADDED LOCATION(S) NODSTS,CONSTA AND REFERENCES IN CWAIT SUBRTN.
51     ; ADDED LOCATION(S) EVTNOD,EVTCOD,EVTCID AND REFERENCES IN EWAIT SUBRTN
52     ; THESE CHANGES WERE COMMENTED OUT.
53

```

```
54
55          SEARCH DFCIBT
56          SALL
57          .DIREC FLBLST
58          NOSYM
59          NAME (DFCIB1,\DECVER,\VEDIT,CTP Connect/Disconnect/Status Subroutines)
60          SEARCH GLXMAC,MONSYM,UUOSYM,MACSYM
61
62          PROLOG (DFCIB1)
63
64          ;Interns
65
66          INTERN STATB,RSCSE,STAT,EVENTB,GEVNT,SCOUNT,NAMED,SCSS$,SCSERC
67          INTERN GLNN,PALLN,DOCON,CONID,SHCST,SHPLL,DODIS,DISC,RDSTIM,OTIME
68          INTERN SEVNT,SCSERR,CONND,MPBUF,UMPBUF,CWAIT,EWAIT
69          INTERN DBUF1,DBUF2,DBUF3,MBUF1,MBUF2,MBUF3
70          INTERN DBUF4,DBUF5,DBUF6,MBUF4,MBUF5,MBUF6,NAMES
71          INTERN SCSERS,RBSIZ,MINMS,MINDS,WAITX,MAPBL,RCONB,CONNB
72          INTERN LNODEN,RCON,GPRS,GPRSB,LNODE,DOCONX,RSPMXM,RSPMXD
73
74          ;Externs found in DFCIBP
75
76          EXTERN MONTYP,NODEN,CHKST,CHKEV,SECS,MAPBBA,RTCHK
77
78          ;Externs found in DFCIB2
79
80          EXTERN BUFLNM,BUFRNM
81
82          ;Externs found in DFCIBM
83
84          EXTERN FLGFLG
```



```

85          SUBTTL  Storage
86
87          ;Message/Datagram Buffers
88
89          000000'      MBUF1:  BLOCK      ^D200
90          000310'      MBUF2:  BLOCK      ^D200
91          000620'      MBUF3:  BLOCK      ^D200
92          001130'      MBUF4:  BLOCK      ^D200
93          001440'      MBUF5:  BLOCK      ^D200
94          001750'      MBUF6:  BLOCK      ^D200
95
96          002260'      DBUF1:  BLOCK      ^D200
97          002570'      DBUF2:  BLOCK      ^D200
98          003100'      DBUF3:  BLOCK      ^D200
99          003410'      DBUF4:  BLOCK      ^D200
100         003720'      DBUF5:  BLOCK      ^D200
101         004230'      DBUF6:  BLOCK      ^D200
102
103         004540' 000000 000000      RSPMXD: 0      ;Responder max datagram size
104         004541' 000000 000000      RSPMXM: 0      ;Responder max message size
105         004542' 000000 000000      LNODEN: 0      ;Local node number saved here
106         004543' 000000 000000      CONID: 0      ;Saved connection id
107         004544' 000000 000000      DISCR: 0      ;Disconnect reason
108         004545' 000000 000000      TIMST: 0      ;Time started
109         004546' 000000 000000      TIMEND: 0     ;Time to end
110         004547' 000000 000000      WAITXE: 0     ;Wait time exhausted (WAITX)
111
112         004550'      SCSCAL: BLOCK      1      ;Remembered SCS caller
113         004551'      SCSERC: BLOCK      1      ;SCS error code from last error
114         004552'      SCSFCN: BLOCK      1      ;Remembered SCS function code
115         004553'      SCSBLK: BLOCK      1      ;Remembered SCS arg block addr
116                                     ;SCSBLK must follow SCSFCN!
117         004554' 000000 000000      SCSERS: 0     ;-1 If an scs error occurred
118         004555' 000000 000000      RSCSE: 0     ;Set to 0 to report a SCS failure
119                                     ;Set to a 1 to not report a SCS failure

```

```

120          SUBTTL Subroutines
121
122          ;*****
123          ;Get and print info about nodes that the monitor knows about.
124          ;*****
125
126 004556' 260 17 0 00 000000*  PALLN:  $SAVE  <P1,P2,P3>      ;Save ACs
127 004557' 402 00 0 00 000010    SETZM   P2          ;Zero P2, keep track of SBI
128
129 004560' 402 00 0 00 000002    PALLN1: SETZM   S2          ;0'S to CID, so all nodes are typed out
130 004561' 200 01 0 00 000010    MOVE     S1,P2        ;Put SBI in S1
131 004562' 260 17 0 00 005272'    $CALL   RCON         ;Go get info about this node
132 004563' 254 00 0 00 004617'    JRST    PALLN2        ;Error occurred
133
134          ;Prepare for massive $TEXT
135
136 004564' 200 01 0 00 005327'    MOVE     S1,RCONB+.SQDSV      ;Get software version
137 004565' 260 17 0 00 004623'    $CALL   P4A8          ;Compute 4 ASCII from 8 bit bytes
138 004566' 200 07 0 00 000002    MOVE     P1,S2          ;Copy it somewhere safe
139 004567' 200 01 0 00 005326'    MOVE     S1,RCONB+.SQDST      ;Get software type
140 004570' 260 17 0 00 004623'    $CALL   P4A8          ;Compute 4 ASCII from 8 bit bytes
141 004571' 200 11 0 00 000002    MOVE     P3,S2          ;Copy it somewhere safe
142 004572' 200 01 0 00 005336'    MOVE     S1,RCONB+.SQNNM      ;Get first 4 bytes of node name
143 004573' 260 17 0 00 004623'    $CALL   P4A8          ;Compute 4 ASCII from 8 bit bytes
144 004574' 200 03 0 00 000002    MOVE     T1,S2          ;Copy it somewhere safe
145 004575' 200 01 0 00 005337'    MOVE     S1,RCONB+.SQNNM+1    ;Get second two bytes of node name
146 004576' 260 17 0 00 004623'    $CALL   P4A8          ;Compute 4 ASCII from 8 bit bytes
147 004577' 200 04 0 00 000002    MOVE     T2,S2          ;Copy it somewhere safe
148 004600' 200 01 0 00 005332'    MOVE     S1,RCONB+.SQDHT      ;Get first 4 bytes of hard type
149 004601' 260 17 0 00 004623'    $CALL   P4A8          ;Compute 4 ASCII from 8 bit bytes
150 004602' 200 05 0 00 000002    MOVE     T3,S2          ;Copy it somewhere safe
151
152 004603' 201 02 0 00 006171'    MOVEI    S2,[ASCII/Illegal/] ;Load default VC state
153 004604' 554 01 0 00 005321'    HLRZ     S1,RCONB+.SQVCS      ;Get virtual cir conn state
154 004605' 306 01 0 00 000000    CAIN     S1,VC.CLO        ;Closed ?
155 004606' 201 02 0 00 006173'    MOVEI    S2,[ASCII/Closed/] ;Yes
156 004607' 306 01 0 00 000001    CAIN     S1,VC.STS        ;Start sent?
157 004610' 201 02 0 00 006175'    MOVEI    S2,[ASCII/Start sent/] ;Yes
158 004611' 306 01 0 00 000002    CAIN     S1,VC.STR        ;Start rec ?
159 004612' 201 02 0 00 006200'    MOVEI    S2,[ASCII/Start received/] ;Yes
160 004613' 306 01 0 00 000003    CAIN     S1,VC.OPN        ;Open ?
161 004614' 201 02 0 00 006203'    MOVEI    S2,[ASCII/Open/] ;Yes

```

```

162
163 $TEXT (,<Node ^D/RCONB+.SQVCS,RHMASK/. Node name: ^T/T1/^T/T2/
164 Virtual circuit state: ^O/RCONB+.SQVCS,LHMASK/, ^T/(S2)/
165 Destination software type: ^T/P3/
166 Destination software version: ^T/P1/
167 Destination hardware type: ^T/T3/
168 Maximum destination datagram size: ^D/RCONB+.SQMDD/.
169 Maximum destination message size: ^D/RCONB+.SQMDM/.
170 Port characteristic word: ^O/RCONB+.SQPCW/>)
171
172 004615' 260 17 0 00 000000*
173 004617' 350 00 0 00 000010
174 004620' 302 10 0 00 000020
175 004621' 254 00 0 00 004560'
176 004622' 263 17 0 00 000000
177
178 ;This subroutine makes 4 left justified ASCII 8 bit bytes into one word ASCII2.
179 ;To use this routine, put the word in S1, the result will be returned in S2.
180 004623' 260 17 0 00 000000*
181 004624' 200 03 0 00 006337'
182 004625' 200 04 0 00 006340'
183 004626' 400 02 0 00 000000
184 004627' 201 06 0 00 000004
185
186 004630' 134 05 0 00 000003
187 004631' 136 05 0 00 000004
188 004632' 367 06 0 00 004630'
189 004633' 263 17 0 00 000000

PALLN2: AOS      P2                ;Bump to next sbi
        CAIE     P2,^D16          ;Tried all SBI's ?
        JRST     PALLN1          ; No, Do next one
        $RET                    ;Return

;This subroutine makes 4 left justified ASCII 8 bit bytes into one word ASCII2.
;To use this routine, put the word in S1, the result will be returned in S2.

P4A8:   $SAVE     <T1,T2,T3,T4>    ;Save scratch ACs
        MOVE      T1,[POINT 8,S1]  ;Load the pointer to the input data
        MOVE      T2,[POINT 7,S2]  ;Load the pointer to the output data
        SETZ      S2,              ;Make ASCII2
        MOVEI     T4,4              ;Load byte count

P4A81:   ILDB      T3,T1             ;Get a byte
        IDPB      T3,T2             ;Store a byte
        SOJG      T4,P4A81          ;Loop for them
        $RET                    ;Return
  
```

```

190
191
192
193
194
195 004634' 304 00 0 00 000000 DOCON: $SAVE <S2,S1> ;Save ACs
196 004641' 201 01 0 00 000310' MOVEI S1,MBUF2 ;Get address of 1st message buffer
197 004642' 202 01 0 00 000000' MOVEM S1,MBUF1 ;Link 1st buffer to second buffer
198 004643' 201 01 0 00 000620' MOVEI S1,MBUF3 ;Get address of 3rd message buffer
199 004644' 202 01 0 00 000310' MOVEM S1,MBUF2 ;Link 2nd buffer to 3rd buffer
200 004645' 201 01 0 00 001130' MOVEI S1,MBUF4 ;Get address of 4th message buffer
201 004646' 202 01 0 00 000620' MOVEM S1,MBUF3 ;Link 4th buffer to 3rd buffer
202 004647' 201 01 0 00 001440' MOVEI S1,MBUF5 ;Get address of 5th message buffer
203 004650' 202 01 0 00 001130' MOVEM S1,MBUF4 ;Link 5th buffer to 4th buffer
204 004651' 201 01 0 00 001750' MOVEI S1,MBUF6 ;Get address of 6th message buffer
205 004652' 202 01 0 00 001440' MOVEM S1,MBUF5 ;Link 6th buffer to 5th buffer
206
207 004653' 402 00 0 00 001750' SETZM MBUF6 ;Mark end of message buffer chain
208
209 004654' 201 01 0 00 002570' MOVEI S1,DBUF2 ;Get address of 1st datagram buffer
210 004655' 202 01 0 00 002260' MOVEM S1,DBUF1 ;Link 1st buffer to second buffer
211 004656' 201 01 0 00 003100' MOVEI S1,DBUF3 ;Get address of 3rd datagram buffer
212 004657' 202 01 0 00 002570' MOVEM S1,DBUF2 ;Link 2nd buffer to 3rd buffer
213 004660' 201 01 0 00 003410' MOVEI S1,DBUF4 ;Get address of 4th datagram buffer
214 004661' 202 01 0 00 003100' MOVEM S1,DBUF3 ;Link 4th buffer to 3rd buffer
215 004662' 201 01 0 00 003720' MOVEI S1,DBUF5 ;Get address of 5th datagram buffer
216 004663' 202 01 0 00 003410' MOVEM S1,DBUF4 ;Link 5th buffer to 4th buffer
217 004664' 201 01 0 00 004230' MOVEI S1,DBUF6 ;Get address of 6th datagram buffer
218 004665' 202 01 0 00 003720' MOVEM S1,DBUF5 ;Link 6th buffer to 5th buffer
219 004666' 402 00 0 00 004230' SETZM DBUF6 ;Mark end of datagram buffer chain
220
221 004667' 201 01 0 00 000000' MOVEI S1,MBUF1 ;Put 1st message buffer address in S1
222 004670' 201 02 0 00 002260' MOVEI S2,DBUF1 ;Put 1st datagram buffer address in S2
223 004671' 260 17 0 00 005154' $CALL CONN ;Do the connection
224 004672' 263 17 0 00 000000' $RET ;Return

```



```

225
226 ;*****
227 ; Make a connection with a node
228 ;*****
229
230 004673' 304 00 0 00 000000 DOCONX: $SAVE <S2,S1> ;Save ACs
231 004700' 201 01 0 00 000310' MOVEI S1,MBUF2 ;GET ADDRESS OF 2ND MSG BUFFER.
232 004701' 202 01 0 00 000000' MOVEM S1,MBUF1 ;Link 1st buffer to second buffer
233 004702' 201 01 0 00 000620' MOVEI S1,MBUF3 ;Get address of 3rd message buffer
234 004703' 202 01 0 00 000310' MOVEM S1,MBUF2 ;Link 2nd buffer to 3rd buffer
235 004704' 201 01 0 00 001130' MOVEI S1,MBUF4 ;Get address of 4th message buffer
236 004705' 202 01 0 00 000620' MOVEM S1,MBUF3 ;Link 4th buffer to 3rd buffer
237 004706' 201 01 0 00 001440' MOVEI S1,MBUF5 ;Get address of 5th message buffer
238 004707' 202 01 0 00 001130' MOVEM S1,MBUF4 ;Link 5th buffer to 4th buffer
239 004710' 201 01 0 00 001750' MOVEI S1,MBUF6 ;Get address of 6th message buffer
240 004711' 202 01 0 00 001440' MOVEM S1,MBUF5 ;Link 6th buffer to 5th buffer
241
242 004712' 402 00 0 00 001750' SETZM MBUF6 ;Mark end of message buffer chain
243
244 004713' 201 01 0 00 002570' MOVEI S1,DBUF2 ;Get address of 1st datagram buffer
245 004714' 202 01 0 00 002260' MOVEM S1,DBUF1 ;Link 1st buffer to second buffer
246 004715' 201 01 0 00 003100' MOVEI S1,DBUF3 ;Get address of 3rd datagram buffer
247 004716' 202 01 0 00 002570' MOVEM S1,DBUF2 ;Link 2nd buffer to 3rd buffer
248 004717' 201 01 0 00 003410' MOVEI S1,DBUF4 ;Get address of 4th datagram buffer
249 004720' 202 01 0 00 003100' MOVEM S1,DBUF3 ;Link 4th buffer to 3rd buffer
250 004721' 201 01 0 00 003720' MOVEI S1,DBUF5 ;Get address of 5th datagram buffer
251 004722' 202 01 0 00 003410' MOVEM S1,DBUF4 ;Link 5th buffer to 4th buffer
252 004723' 201 01 0 00 004230' MOVEI S1,DBUF6 ;Get address of 6th datagram buffer
253 004724' 202 01 0 00 003720' MOVEM S1,DBUF5 ;Link 6th buffer to 5th buffer
254 004725' 402 00 0 00 004230' SETZM DBUF6 ;Mark end of datagram buffer chain
255
256 004726' 201 01 0 00 000000' MOVEI S1,MBUF1 ;Put 1st message buffer address in S1
257 004727' 201 02 0 00 002260' MOVEI S2,DBUF1 ;Put 1st datagram buffer address in S2
258 004730' 260 17 0 00 005154' $CALL CONN ;Do the connection
259
260 004731' 260 17 0 00 005707' $CALL RBSIZ ; Get exerciser buffer sizes.
261 004732' 400 01 0 00 000000' SETZ S1 ; No SBI
262 004733' 200 02 0 00 005212' MOVE S2,CONNB+7 ; S2=CID
263 004734' 260 17 0 00 005272' $CALL RCON ; Retrieve connection data.
264 004735' 200 01 0 00 005316' MOVE S1,RCONB
265 004736' 200 02 0 01 000007' MOVE S2,7(S1) ; Get responder max datagram size
266 004737' 202 02 0 00 004540' MOVEM S2,RSPMXD ; Save at rspmxd
267 004740' 200 02 0 01 000010' MOVE S2,8(S1) ; Get responder max message size
268 004741' 202 02 0 00 004541' MOVEM S2,RSPMXM ; Save at rspmxm
269 004742' 263 17 0 00 000000' $RET

```



```

270
271 ;*****
272 ;Get connection status
273 ;*****
274
275 004743' 304 00 0 00 000000 SHCST: $SAVE <S2,S1> ;Save an AC
276 004750' 260 17 0 00 005252' $CALL STAT ;Get the status on the connection CONID
277 $TEXT (,<Connect ID: ^O/STATB+.SQCID/
Flags (^O/STATB+.SQFST,LHMASK/): ^A>)
278 004751' 260 17 0 00 004615* HLRZ S1,STATB+.SQFST ;Get info
279 004753' 554 01 0 00 005270' MOVE S2,S1 ;Copy it
280 004754' 200 02 0 00 000001 ANDI S2,740000 ;ANDI the flag bits
281 004755' 405 02 0 00 740000 SKIPN S2 ;Skip if non zero
282 004756' 336 00 0 00 000002 $TEXT (,<None^A>) ;No status
283 004757' 260 17 0 00 004751* TRNE S1,400000 ;Bit 0 ?
284 004761' 602 01 0 00 400000 $TEXT (,<Message available flag ^A>)
285 004762' 260 17 0 00 004757* TRNE S1,200000 ;Bit 1 ?
286 004764' 602 01 0 00 200000 $TEXT (,<Datagram available flag ^A>)
287 004765' 260 17 0 00 004762* TRNE S1,100000 ;Bit 2 ?
288 004767' 602 01 0 00 100000 $TEXT (,<DMA transfer complete ^A>)
289 004770' 260 17 0 00 004765* TRNE S1,40000 ;Bit 3 ?
290 004772' 602 01 0 00 040000 $TEXT (,<Event pending flag ^A>)
291 004773' 260 17 0 00 004770*
292

```

```

293
294 004775' 201 02 0 00 006171'      MOVEI S2,[ASCIZ/illegal/]      ;Load description default
295 004776' 550 01 0 00 005270'      HRRZ S1,STATB+.SQFST      ;Get info
296 004777' 306 01 0 00 000001'      CAIN S1,SQ%CL0      ;1 ?
297 005000' 201 02 0 00 006173'      MOVEI S2,[ASCIZ/closed/]      ;Yes
298 005001' 306 01 0 00 000002'      CAIN S1,SQ%LIS      ;2 ?
299 005002' 201 02 0 00 006447'      MOVEI S2,[ASCIZ/Listening/]      ;Yes
300 005003' 306 01 0 00 000003'      CAIN S1,SQ%CSE      ;3 ?
301 005004' 201 02 0 00 006451'      MOVEI S2,[ASCIZ/Connect request was sent/] ;Yes
302 005005' 306 01 0 00 000004'      CAIN S1,SQ%CRE      ;4 ?
303 005006' 201 02 0 00 006456'      MOVEI S2,[ASCIZ/Connect request was received/] ;Yes
304 005007' 306 01 0 00 000005'      CAIN S1,SQ%CAK      ;5 ?
305 005010' 201 02 0 00 006464'      MOVEI S2,[ASCIZ/Connect response was received/] ;Yes
306 005011' 306 01 0 00 000006'      CAIN S1,SQ%ACS      ;6 ?
307 005012' 201 02 0 00 006472'      MOVEI S2,[ASCIZ/Accept request was sent/] ;Yes
308 005013' 306 01 0 00 000007'      CAIN S1,SQ%RJS      ;7 ?
309 005014' 201 02 0 00 006477'      MOVEI S2,[ASCIZ/Reject request was sent/] ;Yes
310 005015' 306 01 0 00 000010'      CAIN S1,SQ%OPN      ;10 ?
311 005016' 201 02 0 00 006504'      MOVEI S2,[ASCIZ/Connection is open/] ;Yes
312 005017' 306 01 0 00 000011'      CAIN S1,SQ%DSE      ;11 ?
313 005020' 201 02 0 00 006510'      MOVEI S2,[ASCIZ/Disconnect request was sent/] ;Yes
314 005021' 306 01 0 00 000012'      CAIN S1,SQ%DRE      ;12 ?
315 005022' 201 02 0 00 006516'      MOVEI S2,[ASCIZ/Disconnect request received/] ;Yes
316 005023' 306 01 0 00 000013'      CAIN S1,SQ%DAK      ;13 ?
317 005024' 201 02 0 00 006524'      MOVEI S2,[ASCIZ/Disconnect response received/] ;Yes
318 005025' 306 01 0 00 000014'      CAIN S1,SQ%DMC      ;14 ?
319 005026' 201 02 0 00 006532'      MOVEI S2,[ASCIZ/Waiting for disconnect response/] ;Yes
320                                $TEXT      (<
321                                Connect state: ^0/STATB+.SQFST/, ^T/(S2)/
322                                Node number: ^D/STATB+.SQSBR,RHMASK/>)
323                                $RET
  
```

```

324
325 ;*****
326 ;Poll the node we connected to
327 ;*****
328
329 005032' 304 00 0 00 000000 SHPOLL: $SAVE <S2,S1> ;Save ACs
330 005037' 260 17 0 00 005117' $CALL POLL ;Poll this connection
331
332 005040' 201 02 0 00 006171' MOVEI S2,[ASCIZ/Illegal/] ;Load default message
333 005041' 200 01 0 00 005147' MOVE S1,POLLB+.SQCST ;Get info
334 005042' 306 01 0 00 000001 CAIN S1,SQXCLO ;1 ?
335 005043' 201 02 0 00 006173' MOVEI S2,[ASCIZ/Closed/] ;Yes
336 005044' 306 01 0 00 000002 CAIN S1,SQXLIS ;2 ?
337 005045' 201 02 0 00 006447' MOVEI S2,[ASCIZ/Listening/] ;Yes
338 005046' 306 01 0 00 000003 CAIN S1,SQXCSE ;3 ?
339 005047' 201 02 0 00 006451' MOVEI S2,[ASCIZ/Connect request was sent/] ;Yes
340 005050' 306 01 0 00 000004 CAIN S1,SQXCRE ;4 ?
341 005051' 201 02 0 00 006456' MOVEI S2,[ASCIZ/Connect request was received/] ;Yes
342 005052' 306 01 0 00 000005 CAIN S1,SQXCAK ;5 ?
343 005053' 201 02 0 00 006464' MOVEI S2,[ASCIZ/Connect response was received/] ;Yes
344 005054' 306 01 0 00 000006 CAIN S1,SQXACS ;6 ?
345 005055' 201 02 0 00 006472' MOVEI S2,[ASCIZ/Accept request was sent/] ;Yes
346 005056' 306 01 0 00 000007 CAIN S1,SQXRJS ;7 ?
347 005057' 201 02 0 00 006477' MOVEI S2,[ASCIZ/Reject request was sent/] ;Yes
348 005060' 306 01 0 00 000010 CAIN S1,SQXOPN ;10 ?
349 005061' 201 02 0 00 006504' MOVEI S2,[ASCIZ/Connection is open/] ;Yes
350 005062' 306 01 0 00 000011 CAIN S1,SQXDSE ;11 ?
351 005063' 201 02 0 00 006510' MOVEI S2,[ASCIZ/Disconnect request was sent/] ;Yes
352 005064' 306 01 0 00 000012 CAIN S1,SQXDRE ;12 ?
353 005065' 201 02 0 00 006516' MOVEI S2,[ASCIZ/Disconnect request received/] ;Yes
354 005066' 306 01 0 00 000013 CAIN S1,SQXDAK ;13 ?
355 005067' 201 02 0 00 006524' MOVEI S2,[ASCIZ/Disconnect response received/] ;Yes
356 005070' 306 01 0 00 000014 CAIN S1,SQXDMC ;14 ?
357 005071' 201 02 0 00 006532' MOVEI S2,[ASCIZ/Waiting for disconnect response/] ;Yes
358 $TEXT (,<Connect ID: ^O/POLLB+.SQCID/
359 Connect state: ^O/POLLB+.SQCST/: ^T/(S2)/
360 Destination connect ID: ^O/POLLB+.SQDCI/
361 Process name: ^T/PRCNAM/
362 Node number of destination: ^D/POLLB+.SQSBI/
363 Source disconnect reasons: ^O/POLLB+.SQREA,LHMASK/
364 005072' 260 17 0 00 005027* Destination disconnect reasons: ^O/POLLB+.SQREA,RHMASK/>)
365 005074' 263 17 0 00 000000 $RET
  
```

366						:*****					
367						;Get and print local node number					
368						:*****					
369											
370											
371	005075'	260	17	0	00	005101'	GLNN:	\$CALL	LNODE	:	Go get the local node number
372	005076'	260	17	0	00	005072*		\$TEXT	(,<The local node number is "D/LNODEN/.>)		
373	005100'	263	17	0	00	000000		\$RET			
374											
375							:*****				
376							;Get local node number				
377							:*****				
378											
379	005101'	304	00	0	00	000000	LNODE:	\$SAVE	<S2,S1>	:	Save some ac's
380	005106'	201	01	0	00	000030		MOVEI	S1,.SSGLN	:	Function code
381	005107'	201	02	0	00	005115'		MOVEI	S2,LNODB	:	Argument block address
382	005110'	260	17	0	00	006051'		\$CALL	SCS\$:	Perform SCS function
383	005111'	260	17	0	00	006107'		\$CALL	SCSERR	:	Print error message
384	005112'	200	01	0	00	005116'		MOVE	S1,LNODB+.SQLNN	:	Get local ci node number
385	005113'	202	01	0	00	004542'		MOVEM	S1,LNODEN	:	Save it
386	005114'	263	17	0	00	000000		\$RET		:	Return to calling
387											
388	005115'	000000			000002		LNODB:	0,...LBGLN		:	Processed words (returned by montr),.
389										:	Length of block (user supplied)
390	005116'	000000			000000			0		:	Local ci node # (returned by montr)

```

391
392
393
394
395
396
397 005117' 304 00 0 00 000000
398 005124' 200 01 0 00 004543'
399 005125' 202 01 0 00 005146'
400 005126' 200 01 0 00 006675'
401 005127' 202 01 0 00 005151'
402 005130' 201 01 0 00 000010'
403 005131' 201 02 0 00 005145'
404 005132' 260 17 0 00 006051'
405 005133' 260 17 0 00 006107'
406 005134' 263 17 0 00 000000
407
408 005135'
409
410 005145' 000000 000007
411
412 005146' 000000 000000
413 005147' 000000 000000
414 005150' 000000 000000
415 005151' 000000 000000
416 005152' 000000 000000
417
418 005153' 000000 000000
419

;*****
;Poll a responder connection
; Call with the cid in in CONID
;*****
POLL:  $SAVE  <S2,S1>          ;Save some ac's
        MOVE  S1,CONID        ;Load the connection ID
        MOVEM S1,POLLB+.SQCID  ;Put the cid in the argument block
        MOVE  S1,[POINT 7,PRCNAM] ;Get byte pntr to dest Process name
        MOVEM S1,POLLB+.SQBDN
        MOVEI S1,.SSCSP        ;Function code
        MOVEI S2,POLLB        ;Argument block address
        $CALL SCSS            ;Perform SCS function
        $CALL SCSERR          ;Print an error message and continue
        $RET                  ;Return to calling

PRCNAM: BLOCK 10                ;Monitor writes destination process

POLLB:  0,...LBCSP              ;Processed words (returned by montr)..
        0                      ; Length of block (user supplied)
        0                      ;Connect id (user supplied)
        0                      ;Connection state (returned by monitor)
        0                      ;Destination connect id (returned by monitor)
        0                      ;Byte pointer to dest process name (user supplied)
        0                      ;Node number of destination (user supplied)
        0,..0                  ;Source disconnect reasons,, dest disconnect reasons

; (Returned by monitor)

```



```

420
421
422
423
424
425
426
427 005154' 202 01 0 00 005210'
428 005155' 202 02 0 00 005211'
429 005156' 304 00 0 00 000000'
430 005163' 402 00 0 00 004543'
431 005164' 201 01 0 00 000010'
432 005165' 202 01 0 00 005203'
433 005166' 514 01 0 00 000000*
434 005167' 202 01 0 00 005206'
435 005170' 200 01 0 00 006676'
436 005171' 202 01 0 00 005204'
437 005172' 200 01 0 00 006677'
438 005173' 202 01 0 00 005205'
439
440 005174' 201 01 0 00 000000'
441 005175' 201 02 0 00 005203'
442 005176' 260 17 0 00 006051'
443 005177' 260 17 0 00 006107'
444 005200' 200 01 0 00 005212'
445 005201' 202 01 0 00 004543'
446 005202' 263 17 0 00 000000'

;*****
;Make a connection
;Call with the node # you want to connect to in NODEN, the address of message
;buffer chain in S1 and the address of datagram buffer chain in S2.
;*****
CONN:  MOVEM  S1,CONNB+.SQAMC      ;Put message chain in argument block
        MOVEM  S2,CONNB+.SQADC      ;Put datagram chain in argument block
        $SAVE  <S2,S1>              ;Save some ac's
        SETZM  CONID                ;Zero saved connection id
        MOVEI  S1,.LBCON             ;Load length of block
        MOVEM  S1,CONNB+.SQLEN      ;Save in arg block
        HRLZ   S1,NODEN              ;Put the dest node # in the left half
        MOVEM  S1,CONNB+.SQSYS      ;Put node address in argument block
        MOVE   S1,[POINT 7,NAMES]   ;Get byte pntr to source process name
        MOVEM  S1,CONNB+.SQSPN      ;Put it in argument block
        MOVE   S1,[POINT 7,NAMED]   ;Get byte pntr to dest process name
        MOVEM  S1,CONNB+.SQDPN      ;Put it in argument block
        MOVEI  S1,.SSCON             ;Function code
        MOVEI  S2,CONNB             ;Argument block address
        $CALL  SCSS$                 ;Perform SCS function
        $CALL  SCSERR                ;Print an error message and continue
        MOVE   S1,CONNB+.SQRCI      ;Get connection id
        MOVEM  S1,CONID              ;Save it
        $RET                          ;Return to calling
    
```

```

447
448 005203' 000000 000010      CONNB: 0...LBCON      ;Processed words (mon returns)...
449                                     ; Length of block (user supplied)
450 005204' 000000 000000      0      ;Byte pntr to source process name (user supply)
451 005205' 000000 000000      0      ;Byte pntr to dest process name (user supplied)
452 005206' 000000 000000      0,0    ;Node #...sysap field (user supplies)
453 005207' 000000 005213'     CONND   ;Address of connection data
454 005210' 000000 000000      0      ;Address of message buffer chain
455 005211' 000000 000000      0      ;Address of datagram buffer chain
456 005212' 000000 000000      0      ;Returned connect id (mon returns)
457
458      ; Connection data block
459
460      000000      COP==0      ;Opcode
461      000003      CVER==3    ;Version
462      000000      CREV==0    ;Revision
463
464 005213' 000 003 000 0000    CONND: BYTE (8) COP,CVER,CREV
465 005214' 000000 000000      0
466 005215' 000000 000000      0
467 005216' 000000 000000      0
468
469 005217' 103 124 120 044 103 NAMES: ASCIZ/CTP$CONTROLLER/
470 005222' 000000 000000      0
471 005223' 000000 000000      0
472 005224' 103 124 120 044 122 NAMED: ASCIZ/CTP$RESPONDER/
473 005227' 000000 000000      0
474 005230' 000000 000000      0
    
```

```

475
476 ;*****
477 ;Disconnect a connection
478 ; Disconnects the CID stored in CONID.
479 ;*****
480
481 005231'
482 005231' 304 00 0 00 000000
483 005236' 200 01 0 00 004543'
484 005237' 202 01 0 00 005250'
485 005240' 201 01 0 00 000003
486 005241' 201 02 0 00 005247'
487 005242' 260 17 0 00 006051'
488 005243' 260 17 0 00 006107'
489 005244' 200 01 0 00 005251'
490 005245' 202 01 0 00 004544'
491 005246' 263 17 0 00 000000
492
493 005247' 000000 000004
494
495 005250' 000000 000000
496 005251' 000000 000000

DODIS:
DISC:  $SAVE  <S2,S1>          ;Save some ac's
        MOVE   S1,CONID        ;Get connect ID
        MOVEM  S1,DISCB+.SQCID ;Put it in the argument block
        MOVEI  S1,.SSDIS       ;Function code
        MOVEI  S2,DISCB        ;Argument block address
        $CALL  SC$             ;Perform SCS function
        $CALL  SCSERR          ;Print an error message and continue
        MOVE   S1,DISCB+.SQDIS ;Get disconnect reason
        MOVEM  S1,DISCR        ;Save it
        $RET                   ;Return to calling

DISCB:  0,...LBDIS+1           ;Processed words (mon returns),,
        0                     ; Length of block (user supplied)
        0                     ;Connect id ( user supplies)
        0                     ;Disconnect reason (monitor returns)
  
```

```

497
498
499
500
501
502
503 005252' 304 00 0 00 000000
504 005257' 200 01 0 00 004543'
505 005260' 202 01 0 00 005267'
506 005261' 201 01 0 00 000012
507 005262' 201 02 0 00 005266'
508 005263' 260 17 0 00 006051'
509 005264' 260 17 0 00 006107'
510 005265' 263 17 0 00 000000
511
512 005266' 000000 000004
513
514 005267' 000000 000000
515 005270' 000000 000000
516 005271' 000000 000000

;*****
;Status info on a connection
; Call with the cid you want the info in CONID
;*****
STAT:  $SAVE  <S2,S1>          ;Save some ac's
        MOVE   S1,CONID        ;Load the connection ID
        MOVEM  S1,STATB+.SQCID ;Put it in the argument block
        MOVEI  S1,SSSTS        ;Function code
        MOVEI  S2,STATB        ;Argument block address
        SCALL  SCSS            ;Perform SCS function
        SCALL  SCSERR          ;Print an error message and continue
        $RET                   ;Return to calling

STATB:  0,...LBSTS             ;Processed words (mon returns),,
        0                     ; Length of block (user supplied)
        0                     ;Connect id ( user supplies)
        0                     ;Flags,,connect state (monitor returns)
        0                     ;Reserved,,# of remote node (monitor returns)

```

```

517
518
519
520
521
522
523
524
525
526
527 005272' 202 02 0 00 005317'
528 005273' 202 01 0 00 005320'
529 005274' 304 00 0 00 000000
530 005301' 336 00 0 00 005317'
531 005302' 476 00 0 00 004555'
532 005303' 201 01 0 00 000024
533 005304' 202 01 0 00 005316'
534 005305' 201 01 0 00 000011
535 005306' 201 02 0 00 005316'
536 005307' 402 00 0 00 004554'
537 005310' 260 17 0 00 006051'
538 005311' 260 17 0 00 006107'
539
540 005312' 336 00 0 00 004554'
541 005313' 350 00 0 17 000000
542
543 005314' 402 00 0 00 004555'
544 005315' 263 17 0 00 000000
545
546 005316' 000000 000024
547
548 005317' 000000 000000
549 005320' 000000 000000
550
551 005321' 000000 000000
552 005322' 000000 000000
553 005323' 000000 000000
554 005324' 000000 000000
555 005325' 000000 000000
556 005326' 000000 000000
557 005327' 000000 000000
558 005330' 000000 000000
559
560 005331' 000000 000000
561 005332' 000000 000000
562 005333' 000000 000000
563 005334' 000000 000000
564 005335' 000000 000000
565 005336' 000000 000000
566 005337' 000000 000000
567 005340' 000000 000000
568
569 005341' 000000 000000
570
571

```

```

*****
;Retrieve connection data
; Call with the non zero cid you want the info to in S2, or 0 in S2 and
; the SBI number you want to read in S1.
; The routine will return +1 on an error or +2 if no SCS error occurred.
; If the CID was supplied the SCS error is reported. If the SBI was
; supplied the SCS error is not reported.
*****
RCON:  MOVEM  S2,RCONB+.SQCID      ;Put connect id in the argument block
        MOVEM  S1,RCONB+.SQOSB    ;Put SBI in the argument block
        $SAVE  <S2,S1>            ;Save some ac's
        SKIPN  RCONB+.SQCID        ;Skip if CID is supplied
        SETOM  RSCSE               ;Set flag word to not report SCS error
        MOVEI  S1,.LBRCD           ;Load length of the block
        MOVEM  S1,RCONB+.SQLEN     ;Save it here
        MOVEI  S1,.SSRCD           ;Function code
        MOVEI  S2,RCONB            ;Argument block address
        SETZM  SCSERS              ;Clear SCS failure indication
        $CALL  SCS$                ;Perform SCS function
        $CALL  SCSERR              ;Print an error message and continue

        SKIPN  SCSERS              ;Skip if a SCS failure occurred
        AOS    (P)                 ;Bump to good return

RCON4:  SETZM  RSCSE                ;Set flag word to report SCS error
        $RET                          ;Return to calling

RCONB:  0,...LBRCD                 ;Processed words (mon returns),,
                                     ; Length of block (user supplied)
                                     ;Optional connect id ( user supplies)
                                     ;Optional system block index (user sup)

;Returned by monitor
0,..0
;Virtual circuit state,,PORT number
;System address (6 bytes 8 BIT BYTES-
; WORD ALIGNED)
;Maximum destination datagram size
;Maximum destination message size
;Destination software type
;Destination software version
;Destination software edit level

;Destination hardware type

;Destination hardware version
;Destination PORT NAME, 3 WORDS
;..
;Port characteristic, 2 words
;
;Local port number
;SAFETY BUFFER

```


DFCIB1 CTP Connect/Disconnect/Status Subroutines version 2(20) MACRO %53B(1242) 17:32 27-Aug-85 Page 17-1
DFCIB1 MAC 27-Aug-85 11:17 Subroutines

SEQ 0160

572 005342'

BLOCK 10

```

573
574 ;'SHOW EVENT QUEUE'. Call via '%CALL SEVNT'.
575 ;This routine uses the connect ID stored in CONID.
576
577 005352' 304 00 0 00 000000 SEVNT: $SAVE <S2,S1> ;Save some ac's
578 005357' 476 00 0 00 004555' SETOM RSCSE ;Don't report SCS error
579 005360' 260 17 0 00 005423' $CALL GEVNT ;Go get event status
580 005361' 402 00 0 00 004555' SETZM RSCSE ;No errors
581 005362' 332 00 0 00 004554' SKIPE SCSERS ;Skip if no SCS failure occurred
582 JRST [$TEXT (,<? No entry on event queue>)] ;Go exit this routine
583 005363' 254 00 0 00 006712' $RET]
584
585 005364' 201 02 0 00 006171' MOVEI S2,[ASCIZ/Illegal/] ;Load default type
586 005365' 200 01 0 00 005453' MOVE S1,EVENTB+.SQEVT ;Get info
587 005366' 306 01 0 00 000001' CAIN S1,.SEVCC ;Skip if not equal
588 005367' 201 02 0 00 006715' MOVEI S2,[ASCIZ/VC broken/] ;Yes
589 005370' 306 01 0 00 000002' CAIN S1,.SECTL ;Skip if not equal
590 005371' 201 02 0 00 006717' MOVEI S2,[ASCIZ/Connect to listen/] ;Yes
591 005372' 306 01 0 00 000003' CAIN S1,.SECRA ;Skip if not equal
592 005373' 201 02 0 00 006723' MOVEI S2,[ASCIZ/Connection was accepted/] ;Yes
593 005374' 306 01 0 00 000004' CAIN S1,.SECRR ;Skip if not equal
594 005375' 201 02 0 00 006730' MOVEI S2,[ASCIZ/Connection was rejected/] ;Yes
595 005376' 306 01 0 00 000005' CAIN S1,.SEMSC ;Skip if not equal
596 005377' 201 02 0 00 006735' MOVEI S2,[ASCIZ/Message or datagram send complete/] ;Yes
597 005400' 306 01 0 00 000006' CAIN S1,.SELCL ;Skip if not equal
598 005401' 201 02 0 00 006744' MOVEI S2,[ASCIZ/Little credit left/] ;Yes
599 005402' 306 01 0 00 000007' CAIN S1,.SENWO ;Skip if not equal
600 005403' 201 02 0 00 006750' MOVEI S2,[ASCIZ/Node went offline/] ;Yes
601 005404' 306 01 0 00 000010' CAIN S1,.SENCO ;Skip if not equal
602 005405' 201 02 0 00 006754' MOVEI S2,[ASCIZ/Node came online/] ;Yes
603 005406' 306 01 0 00 000011' CAIN S1,.SEOSD ;Skip if not equal
604 005407' 201 02 0 00 006760' MOVEI S2,[ASCIZ/OK to send data/] ;Yes
605 005410' 306 01 0 00 000012' CAIN S1,.SERID ;Skip if not equal
606 005411' 201 02 0 00 006764' MOVEI S2,[ASCIZ/Remote initiated disconnect/] ;Yes
607 005412' 306 01 0 00 000013' CAIN S1,.SEPBC ;Skip if not equal
608 005413' 201 02 0 00 006772' MOVEI S2,[ASCIZ/Port broke connection/] ;Yes
609 005414' 306 01 0 00 000014' CAIN S1,.SECIA ;Skip if not equal
610 005415' 201 02 0 00 006777' MOVEI S2,[ASCIZ/Credit is available/] ;Yes
611 005416' 306 01 0 00 000015' CAIN S1,.SEMDC ;Skip if not equal
612 005417' 201 02 0 00 007003' MOVEI S2,[ASCIZ/Maint data xfer complete/] ;Yes
613 $TEXT (,<Connect id: ^O/EVENTB+.SQCID/
614 Reserved: ^O/EVENTB+.SQESB,LHMASK/ SBI of remote: ^O/EVENTB+.SQESB,RHMASK/
615 Event code: ^O/EVENTB+.SQEVT/: ^T/(S2)/
616 Event data word 1: ^O/EVENTB+.SQDTA/
617 Event data word 2: ^O/EVENTB+.SQDTA+1/
618 Event data word 3: ^O/EVENTB+.SQDTA+2/
619 005420' 260 17 0 00 005076* Event data word 4: ^O/EVENTB+.SQDTA+3/>)
620 005422' 260 17 0 00 000000 $RET

```

```

621
622 ;Get event status. Call via '$CALL GEVNT'.
623 ;This routine uses the CONNECT ID stored in CONID.
624
625 005423' 304 00 0 00 000000 GEVNT: $SAVE <S2,S1> ;Save some ac's
626 005430' 200 01 0 00 004543' MOVE S1,CONID ;Get the connect id
627 005431' 202 01 0 00 005451' MOVEM S1,EVENTB+.SQCID ;Put connect id in the argument block
628 005432' 201 01 0 00 000010 MOVEI S1,.LBEVT ;Get length of argument block
629 005433' 202 01 0 00 005450' MOVEM S1,EVENTB+.SQLEN ;Put it in the argument block
630 005434' 402 00 0 00 005452' SETZM EVENTB+.SQESB ;Zero SBI argument entry
631 005435' 402 00 0 00 005453' SETZM EVENTB+.SQEVT ;Zero SBI argument entry
632 005436' 402 00 0 00 005454' SETZM EVENTB+.SQDTA ;Zero SBI argument entry
633 005437' 402 00 0 00 005455' SETZM EVENTB+.SQDTA+1 ;Zero SBI argument entry
634 005440' 402 00 0 00 005456' SETZM EVENTB+.SQDTA+2 ;Zero SBI argument entry
635 005441' 402 00 0 00 005457' SETZM EVENTB+.SQDTA+3 ;Zero SBI argument entry
636
637 005442' 201 01 0 00 000025 MOVEI S1,.SSEVT ;Function code
638 005443' 201 02 0 00 005450' MOVEI S2,EVENTB ;Argument block address
639 005444' 402 00 0 00 004554' SETZM SCSERS ;Clear SCS failure indication
640 005445' 260 17 0 00 006051' $CALL SCSS ;Perform SCS function
641 005446' 260 17 0 00 006107' $CALL SCSERR ;Print an error message and continue
642 005447' 263 17 0 00 000000 $RET
643
644 005450' 000000 000000 EVENTB: 0 ;Length of block
645 005451' 000000 000000 0 ;Connect ID
646 005452' 000000 000000 0 ;SBI of remote
647 005453' 000000 000000 0 ;Event code
648 005454' 000000 000000 0 ;Event data
649 005455' 000000 000000 0 ;
650 005456' 000000 000000 0 ;
651 005457' 000000 000000 0 ;
    
```

```

652
653 ;*****
654 ;Wait for a STATUS-OF-CONNECTION state or flag to occur.
655 ;Put the state or flag word, but only 1 at at time in location CHKST
656 ;and call this routine via '$CALL CWAIT'. This routine will return when
657 ;the requested state or flag occurs or when the timeout occurs.
658 ;A message will be printed if the timeout occurs.
659 ;*****
660
661 005460' 304 00 0 00 000000 CWAIT: $SAVE <S2,S1> ;Save some ACs
662
663 005465' 260 17 0 00 005572' $CALL RDSTIM ;Read start time
664 005466' 200 01 0 00 000000* MOVE S1,CHKST ;Get state or flag word to test
665 005467' 405 01 0 00 777777 ANDI S1,-1 ;Just look at STATE field
666 005470' 306 01 0 00 000000 CAIN S1,0 ;Skip if state bit supplied
667 005471' 254 00 0 00 005503' JRST CWAIT2 ; No, want to test flag field
668
669 ;Test state field
670
671 005472' 260 17 0 00 000000* CWAIT1: $CALL RTCHK ;Check runtime status
672 005473' 260 17 0 00 005252' $CALL STAT ;Get the status on the connection CONID
673 005474' 200 02 0 00 005270' MOVE S2,STATB+.SQFST ;Get connection state
674 005475' 405 02 0 00 000017 ANDI S2,17 ;Just look at connection state field
675 005476' 316 02 0 00 000001 CAMN S2,S1 ;Is this the state being looked for ?
676 005477' 254 00 0 00 005517' JRST CWAIT8 ; Yes
677 005500' 260 17 0 00 005604' $CALL OTIME ;See if timeout has expired
678 005501' 254 00 0 00 005516' JRST CWAIT7 ; Yes, error
679 005502' 254 00 0 00 005472' JRST CWAIT1 ; No, try again
680
681 ;Test flag field
682
683 005503' 200 01 0 00 005466* CWAIT2: MOVE S1,CHKST ;Get state or flag to test
684 005504' 404 01 0 00 007107' AND S1,[-1,.0] ;Just look at flag field
685
686 005505' 260 17 0 00 005472* CWAIT3: $CALL RTCHK ;Check runtime status
687 005506' 260 17 0 00 005252' $CALL STAT ;Get the status on the connection CONID
688 005507' 200 02 0 00 005270' MOVE S2,STATB+.SQFST ;Get connection state
689 005510' 404 02 0 00 000001 AND S2,S1 ;Just look at connection flag field bit
690 005511' 316 02 0 00 000001 CAMN S2,S1 ;Is this the flag being looked for ?
691 005512' 254 00 0 00 005517' JRST CWAIT8 ; Yes
692 005513' 260 17 0 00 005604' $CALL OTIME ;See if the timeout has expired
693 005514' 254 00 0 00 005516' JRST CWAIT7 ; Yes, error
694 005515' 254 00 0 00 005505' JRST CWAIT3 ; No, try again
695
696 005516' 334 00 0 00 000000 CWAIT7: SKIPA ;Don't bump return address
697 005517' 350 00 0 17 000000 CWAIT8: AOS (P) ;Bump return address for good return
698 005520' 263 17 0 00 000000 $RET

```



```

699
700 ;*****
701 ;Wait for a EVENT-STATUS code to occur.
702 ;Put the event code in location CHKEV and call this routine via
703 ; '$CALL EWAIT'. This routine will return when the requested event status
704 ; occurs or when the timeout occurs. A message will be printed
705 ; if the timeout occurs.
706 ;*****
707
708 005521' 304 00 0 00 000000 EWAIT: $SAVE <S2,S1> ;Save some ACs
709 005526' 200 01 0 00 000000* MOVE S1,CHKEV ;Get event code to test
710 005527' 260 17 0 00 005423' $CALL GEVNT ;Get the event queue of the connection
711 005530' 332 00 0 00 004554' SKIPE SCSERS ;Skip if no SCS failure occurred
712 005531' 254 00 0 00 005536' JRST EWAIT7 ; SCS failure occurred, do error exit
713 005532' 200 02 0 00 005453' MOVE S2,EVENTB+.SQEVT ;Get event code
714 005533' 405 02 0 00 000017 ANDI S2,17 ;Just look at event code limit of field
715 005534' 316 02 0 00 000001 CAMN S2,S1 ;This the event code being looked for ?
716 005535' 254 00 0 00 005537' JRST EWAIT8 ; Yes
717
718 005536' 334 00 0 00 000000 EWAIT7: SKIPA ;Don't bump return address
719
720 005537' 350 00 0 17 000000 EWAIT8: AOS (P) ;Bump return address for good return
721 005540' 263 17 0 00 000000 $RET
722
723 ;Routine to get the uptime in milliseconds to S1
724
725 005541' 336 00 0 00 000000* GETTIM: SKIPN MONTYP ;TOPS-20?
726 005542' 254 00 0 00 005545' JRST GETT11 ;No
727 005543' 104 00 0 00 000014 TIME% ;Get uptime in S1
728 005544' 263 17 0 00 000000 $RET ;Return
729
730 005545' 304 00 0 00 000000 GETT11: $SAVE <S2> ;Save S2
731 005551' 336 00 0 00 005571' SKIPN CYCL60 ;Computed 60HZ flag yet?
732 005552' 260 17 0 00 005561' $CALL GETT12 ;No, get it now
733 005553' 200 01 0 00 007115' MOVE S1,[136,,11] ;System uptime in jiffies %CNSUP
734 005554' 047 01 0 00 000041 GETTAB S1, ;or CALLI S1,41
735 005555' 400 01 0 00 000000 SETZ S1, ;not likely it will fail
736 005556' 221 01 0 00 001750 IMULI S1,^D1000 ;convert to milli-jiffies
737 005557' 230 01 0 00 005571' IDIV S1,CYCL60 ;Convert to milli-seconds
738 005560' 263 17 0 00 000000 $RET ;Return
739
740 005561' 200 01 0 00 007116' GETT12: MOVE S1,[17,,11] ;Load %CNSTS system flags
741 005562' 047 01 0 00 000041 GETTAB S1, ;or CALLI S1,41
742 005563' 400 01 0 00 000000 SETZ S1, ;should not execute this, assume 60
743 005564' 603 01 0 00 004000 TLNE S1,(1B6) ;Is ST%CYC set for 50Hz?
744 005565' 334 01 0 00 007117' SKIPA S1,[^D50] ;Yes, this must be Europe mon ami
745 005566' 201 01 0 00 000074 MOVEI S1,^D60 ;No, this is anywhere else
746 005567' 202 01 0 00 005571' MOVEM S1,CYCL60 ;Set divisor
747 005570' 263 17 0 00 000000 $RET ; and return
748
749 005571' CYCL60: BLOCK 1 ;60. or 50. jiffies per second

```


[illegible]

```

795
796
797
798
799
800
801
802 005632' 304 00 0 00 000000
803 005637' 402 00 0 00 005665'
804 005640' 200 01 0 00 005662'
805 005641' 202 01 0 00 005666'
806 005642' 200 01 0 00 000000*
807 005643' 202 01 0 00 005667'
808 005644' 201 01 0 00 000004'
809 005645' 202 01 0 00 005664'
810
811
812
813 005646' 336 00 0 00 000000*
814 005647' 254 00 0 00 005653'
815 005650' 200 01 0 00 005664'
816 005651' 435 01 0 00 000010'
817 005652' 202 01 0 00 005664'
818
819 005653' 201 01 0 00 000014'
820 005654' 201 02 0 00 005663'
821 005655' 260 17 0 00 006051'
822 005656' 260 17 0 00 006107'
823 005657' 200 01 0 00 005665'
824 005660' 202 01 0 00 000000*
825 005661' 260 17 0 00 000000
826
827 005662' 000000 000000
828
829 005663' 000000 000005
830
831 005664' 000000 000000
832 005665' 000000 000000
833
834
835
836 005666' 000000 000000
837 005667' 000000 000000
838 005670' 000000 000000

;*****
;Map a buffer.
; Call via '$CALL MPBUF'. This subroutine maps a 1k buffer.
; The returned buffer name is stored in 'BUFLNM'.
;*****

MPBUF: $SAVE <S2,S1> ;Save some ac's
      SETZM MAPB+.SQBNA ;Zero returned buffer name
      MOVE S1,MAPBL ;Get length of buffer to map
      MOVEM S1,MAPB+.SQBNA+1 ;Put it in the argument block
      MOVE S1,MAPBBA ;Load address of mapbb
      MOVEM S1,MAPB+.SQBNA+2 ;Save it here
      MOVEI S1,4 ; WRITE ENABLE IN FLAG
      MOVEM S1,MAPB+.SQXFL

;SET 'NOT CLEAR BIT IN FLAG WORD'

      SKIPN FLGFLG ;Skip if flag word set
      JRST MPBUF1 ;Nope
      MOVE S1,MAPB+.SQXFL ;Get word to set
      ORI S1,SQXCVD ;Set the bit
      MOVEM S1,MAPB+.SQXFL ;Save it

MPBUF1: MOVEI S1,SSMAP ;Function code
      MOVEI S2,MAPB ;Argument block address
      $CALL SCSS ;Perform SCS function
      $CALL SCSERR ;Print an error message and continue
      MOVE S1,MAPB+.SQBNA ;Get returned buffer name
      MOVEM S1,BUFLNM ;Save it in CIE1.MAC
      $RET ;Return to calling

MAPBL: 0 ;Supplied by CITSTP.MAC

MAPB: 0,,5 ;Processed words (mon returns),,
          ; Length of block (user supplied)
          0 ;Flags = Industry compatible mode (user supply)
          0 ;Returned buffer name

;Buffer length and address pairs

          0 ;Length in bytes
          0 ;Address of buffer segment
          0 ;Just a extra 0 for no good reason

```

```

839
840 ;*****
841 ;Unmap a buffer
842 ; Call via '%CALL UMPBF'. This subroutine unmaps the buffer name generated by
843 ; by the MAPBUF subroutine
844 ;*****
845
846 005671' 304 00 0 00 000000 UMPBUF: $SAVE <S2,S1> ;Save some ac's
847 005676' 200 01 0 00 005660* MOVE S1,BUFRNM ;Get buffer name
848 005677' 202 01 0 00 005706' MOVEM S1,UMAPB+.SQNAM ;Put buffer name in control block
849 005700' 201 01 0 00 000015' MOVEI S1,SSUMP ;Function code
850 005701' 201 02 0 00 005705' MOVEI S2,UMAPB ;Argument block address
851 005702' 260 17 0 00 006051' $CALL SCSS ;Perform SCS function
852 005703' 260 17 0 00 006107' $CALL SCSERR ;Print an error message and continue
853 005704' 263 17 0 00 000000 $RET ;Return to calling
854
855 005705' 000000 000002 UMAPB: 0,,2 ;Processed words (mon returns)..
856 ; Length of block (user supplied)
857 005706' 000000 000000 0 ;Buffer name

```

```

858
859
860
861
862
863
864 005707' 304 00 0 00 000000
865 005714' 201 01 0 00 000035
866 005715' 201 02 0 00 005725'
867 005716' 260 17 0 00 006051'
868 005717' 260 17 0 00 006107'
869 005720' 200 01 0 00 005726'
870 005721' 202 01 0 00 005730'
871 005722' 200 01 0 00 005727'
872 005723' 202 01 0 00 005731'
873 005724' 263 17 0 00 000000
874
875 005725' 000000 000003
876
877 005726' 000000 000000
878 005727' 000000 000000
879
880 005730' 000000 000000
881 005731' 000000 000000

;*****
;Return buffer sizes
;Call via '%CALL RBSIZ'
;*****

RBSIZ: $SAVE <S2,S1> ;Save some ac's
        MOVEI S1,SSRBS ;Function code
        MOVEI S2,RBSIZB ;Argument block address
        $CALL SCSS ;Perform SCS function
        $CALL SCSERR ;Print an error message and continue
        MOVE S1,RBSIZB+.SQLMG ;Get message size
        MOVEM S1,MINMS ;Save it
        MOVE S1,RBSIZB+.SQLDG ;Get datagram size
        MOVEM S1,MINDS ;Save it
        $RET ;Return to calling

RBSIZB: 0,...LBRBS ;Processed words (returned by montr)..
          0 ;Length of block (user supplied)
          0 ;Minimum Length in words of message buffers
          0 ;Minimum Length in words of datagram buffers

MINMS: 0 ;Minimum message size
MINDS: 0 ;Minimum datagram size

```

```

882
883
884
885
886
887
888
889
890 005732' 304 00 0 00 000000
891 005741' 336 00 0 00 005541*
892 005742' 254 00 0 00 005755'
893 005743' 104 00 0 00 000530
894 005744' 320 16 0 00 005746'
895 005745' 263 17 0 00 000000
896 005746' 550 02 0 17 000000
897 005747' 275 02 0 00 000001
898 005750' 260 17 0 00 005420*
899
900 005752' 260 17 0 00 005750*
901 005754' 263 17 0 00 000000
902
903 005755' 047 01 0 00 000163
904 005756' 304 00 0 00 000000
905 005757' 263 17 0 00 000000
906 005760' 260 17 0 00 005752*
907 005762' 254 00 0 00 005752'
908
909 005763' 000 00 0 00 000000
910 005764' 000 00 0 00 000000
911 005765' 000 00 0 00 000000
912 005766' 000 00 0 00 000000
913 005767' 000 00 0 00 000000
914 005770' 000 00 0 00 000000
915 005771' 000 00 0 00 000000
916 005772' 000 00 0 00 000000
917 005773' 000 00 0 00 000000
918 005774' 000 00 0 00 000000
919 005775' 000 00 0 00 000000
920 005776' 000 00 0 00 000000
921 005777' 000 00 0 00 000000
922 006000' 000 00 0 00 000000
923 006001' 000 00 0 00 000000
924 006002' 000 00 0 00 000000
925 006003' 000 00 0 00 000000
926 006004' 000 00 0 00 000000
927 006005' 000 00 0 00 000000
928 006006' 000 00 0 00 000000
929 006007' 000 00 0 00 000000
930 006010' 000 00 0 00 000000
931

```



```

932
933 ;Here to store the counters
934
935 006011' 304 00 0 00 000000 SCOUNT: $SAVE <S1,S2> ;Save ac's
936
937 006016' 201 01 0 00 000107 MOVEI S1,107 ;Store the 'COUNTERS'
938 006017' 202 01 0 00 005763' MOVEM S1,DIAGAG ; Function code
939 006020' 200 01 0 00 007176' MOVE S1,[7,,3] ;Port number in left half
940 ;Read counters function code in right
941 006021' 202 01 0 00 005764' MOVEM S1,DIAGAG+1 ;Store in arg block
942 006022' 402 00 0 00 005765' SETZM DIAGAG+2 ;Zero rest of the
943 006023' 402 00 0 00 005766' SETZM DIAGAG+3 ; Arg block
944
945 006024' 200 01 0 00 007177' MOVE S1,[^-D21,,DIAGAG] ;Neg length of arg block in left half
946 ;Address of arg block in right half
947 006025' 260 17 0 00 005732' $CALL DIAGJ ;Do the 'DIAG' jsys
948
949 $TEXT (,<^H/[-1]/
950 Microcode version: ^D/DIAGAG+3/
951 Path A: ack count: ^D/DIAGAG+4/.
952 nak count: ^D/DIAGAG+5/.
953 no response count: ^D/DIAGAG+6/.
954 Path B: ack count: ^D/DIAGAG+7/.
955 nak count: ^D/DIAGAG+^D8/.
956 no response count: ^D/DIAGAG+^D9/.
957 Number of dropped datagrams: ^D/DIAGAG+^D10/.
958 Number of packets transmitted: ^D/DIAGAG+^D11/.
959 Number of packets received: ^D/DIAGAG+^D12/.
960 Designated port for counters: ^D/DIAGAG+^D13/.
961 Mover par pre errors: ^D/DIAGAG+^D14,LHMASK/. CBUS par errors: ^D/DIAGAG+^D14,RHMASK/.
962 Reg plipe errors: ^D/DIAGAG+^D15,LHMASK/. Data plipe errors: ^D/DIAGAG+^D15,RHMASK/.
963 Channel errors: ^D/DIAGAG+^D16,LHMASK/. EBUS par plipe errors: ^D/DIAGAG+^D16,RHMASK/.
964 Spurious channel errors: ^D/DIAGAG+^D17,LHMASK/. CBUS Avail Timeouts: ^D/DIAGAG+^D17,RHMASK/.
965
966 Spurious rcv attentions: ^D/DIAGAG+^D18,LHMASK/. Spurious xmit attentions: ^D/DIAGAG+^D18,RHMASK/.
967
968 006026' 260 17 0 00 005760* Xmit buff par errors: ^D/DIAGAG+^D19,LHMASK/. Transmit timeouts: ^D/DIAGAG+^D19,RHMASK/.)
969 006030' 263 17 0 00 000000 $RET

```

```

970
971 ;*****
972 ;Get the return path status for a node. Call this subroutine with the node
973 ;number of the node in S1. Call via '$CALL GPRS'. The return path status will
974 ;be available in the argument block, GPRSB.
975 ;*****
976
977 006031' 202 01 0 00 006047' GPRS: MOVEM S1,GPRSB+.SQRPN ;Put the node number in argument block
978 006032' 304 00 0 00 000000 $SAVE <S2,S1> ;Save some ACs
979 006037' 402 00 0 00 006050' SETZM GPRSB+.SQRPS ;Zero the last returned path status
980 006040' 402 00 0 00 004554' SETZM SCSERS ;Clear SCS failure indication
981 006041' 201 01 0 00 000036 MOVEI S1,.SSRPS ;Function code
982 006042' 201 02 0 00 006046' MOVEI S2,GPRSB ;Argument block address
983 006043' 260 17 0 00 006051' $CALL SCS$ ;Perform SCS function
984 006044' 260 17 0 00 006107' $CALL SCSERR ;Print an error message and continue
985 006045' 263 17 0 00 000000 $RET ;Return to the code that called this
986
987 ;.SSRPS argument block
988
989 006046' 000000 000003 GPRSB: .SQRPS+1 ;Length of block
990 006047' 000 00 0 00 000000 Z ;Node number
991 006050' 000 00 0 00 000000 Z ;Path status
  
```

```

992          SUBTTL Perform SCS Functions
993
994          ;This routine performs an SCS JSYS or UUD depending on the operating system.
995          ;Call: MOVEI S1,function
996          ;       MOVEI S2,argblk
997          ;       $CALL SCS$
998          ;       return+1 error
999          ;       return+2 success
1000
1001 006051' 124 01 0 00 004552' SCS$: DMOVEM S1,SCSFCN ;Save SCS function code, arg block adr
1002 006052' 250 02 0 17 000000 EXCH S2,(P) ;Get return PC, save S2
1003 006053' 552 02 0 00 004550' HRRZM S2,SCSCAL ;Save calling PC
1004 006054' 250 02 0 17 000000 EXCH S2,(P) ;Save return PC, get S2
1005 006055' 336 00 0 00 005741* SKIPN MONTYP ;Skip if TOPS-20
1006 006056' 254 00 0 00 006067' JRST SCS$1 ;TOPS-10
1007
1008          ;Here if TOPS-20
1009
1010 006057' 104 00 0 00 000622 SCS% ;Do the JSYS
1011 006060' 320 16 0 00 006063' ERJMP SCS$2 ;Error, return+1
1012 006061' 350 00 0 17 000000 AOS (P) ;Success, return+2
1013 006062' 263 17 0 00 000000 $RET ;Return
1014
1015 006063' 201 01 0 00 400000 SCS$2: MOVEI S1,..FHSLF ;Load this fork handle
1016 006064' 104 00 0 00 000012 GETER% ;Get last error code
1017 006065' 552 02 0 00 004551' HRRZM S2,SCSERC ;Store error code
1018 006066' 263 17 0 00 000000 $RET ;Return
1019
1020          ;Here for TOPS-10. The SCS UUD wants length,,functiocode in the first word
1021          ; of the block were the SCS JSYS wants 0,,length. Swap the arguments around.
1022
1023 006067' 504 01 0 02 000000 SCS$1: HRL S1,(S2) ;Get length, functioncode
1024 006070' 202 01 0 02 000000 MOVEM S1,(S2) ;Save it back in the argument block
1025 006071' 200 01 0 03 000002 MOVE S1,S2 ;Copy the address of the block
1026 006072' 047 01 0 00 000213 SCS S1, ;The UUD
1027 006073' 304 00 0 00 000000 CAIA ;Error return
1028 006074' 350 00 0 17 000000 AOS (P) ;Give skip return
1029 006075' 202 01 0 00 004551' MOVEM S1,SCSERC ;Save error code
1030 006076' 302 01 0 00 000013 CAIE S1,SSIC1% ;Invalid connect ID?
1031 006077' 263 17 0 00 000000 $RET ;No, return
1032 006100' 120 01 0 00 004552' DMOVE S1,SCSFCN ;Load the function code
1033 006101' 302 01 0 00 000012 CAIE S1,..SSSTS ;Was it get status?
1034 006102' 263 17 0 00 000000 $RET ;No, return
1035 006103' 201 01 0 00 000001 MOVX S1,SQ%CL0 ;Load closed state
1036 006104' 202 01 0 02 000002 MOVEM S1,.SQFST(S2) ;Store this as the state
1037 006105' 350 00 0 17 000000 AOS (P) ;Bump return addr
1038 006106' 263 17 0 00 000000 $RET ;Return

```

```

1039
1040 ;Come here when an error occurs (SCS$ gives nonskip return).
1041 ; Set the contents of RSCSE to nonzero to not report a SCS failure.
1042 ; This routine sets RSCSE to 0 after execution.
1043
1044 006107' 476 00 0 00 004554' SCSERR: SETOM SCSERS ;Mark that a scs error has occurred
1045 006110' 332 00 0 00 004555' SKIPE RSCSE ;Skip if wanted to report SCS error
1046 006111' 254 00 0 00 006131' JRST SCSER3 ; Don't report SCS error
1047 006112' 336 00 0 00 006055* SKIPN MONTYP ;Skip if TOPS-20
1048 006113' 254 00 0 00 006117' JRST SCSER1 ;TOPS-10
1049 006114' 260 17 0 00 006026* $TEXT (,<? SCS JSYS error: ^E/SCSERC/>)
1050 006116' 254 00 0 00 006126' JRST SCSER4 ;Continue below
1051
1052 006117' 205 02 0 00 777743 SCSER1: MOVSJ S2,-ERRSCL ;Load -ive length of table,,0
1053 006120' 554 01 0 02 006133' SCSER2: HLRZ S1,ERRSCT(S2) ;Load error code
1054 006121' 312 01 0 00 004551' CAME S1,SCSERC ;Match?
1055 006122' 253 02 0 00 006120' AOBJN S2,SCSER2 ;Nope, loop
1056 006123' 550 02 0 02 006133' HRRZ S2,ERRSCT(S2) ;Load address of ASCIIZ text
1057 006124' 260 17 0 00 006114* $TEXT (,<? SCS UUO error: (^O/SCSERC/) ^T/(S2)/>)
1058
1059 006126' 370 00 0 00 004550' SCSER4: SOS SCSCAL ;Set true return PC
1060 $TEXT (,< Function ^O/SCSFCN/, argument block at ^O/SCSBLK/, called from PC ^O/SC
1061 006127' 260 17 0 00 006124* SCAL/>)
1062 006131' 402 00 0 00 004555' SCSER3: SETZM RSCSE ;Reset RSCSE to report SCS errors
1063 006132' 263 17 0 00 000000 $RET ;Return to calling
  
```


1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097

006133' 000000 007556'
 006134' 000001 007561'
 006135' 000002 007566'
 006136' 000003 007573'
 006137' 000004 007602'
 006140' 000005 007610'
 006141' 000006 007615'
 006142' 000007 007623'
 006143' 000010 007631'
 006144' 000011 007636'
 006145' 000012 007641'
 006146' 000013 007646'
 006147' 000014 007652'
 006150' 000015 007656'
 006151' 000016 007666'
 006152' 000017 007673'
 006153' 000020 007701'
 006154' 000021 007710'
 006155' 000022 007715'
 006156' 000023 007722'
 006157' 000024 007726'
 006160' 000025 007733'
 006161' 000026 007737'
 006162' 000027 007746'
 006163' 000030 007752'
 006164' 000031 007756'
 006165' 000032 007766'
 006166' 000033 007772'
 006167' 000034 010000'
 006170' 777777 010010'
 000035

;Table of SCS UUD error codes

ERRSCT: XWD SSNPV%,[ASCIZ/Not privileged/]
 XWD SSIFC%,[ASCIZ/Illegal function code/]
 XWD SSARG%,[ASCIZ/Bad argument list length/]
 XWD SSACR%,[ASCIZ/Address check reading arguments/]
 XWD SSACS%,[ASCIZ/Address check storing data/]
 XWD SSCPN%,[ASCIZ/CPU number out of range/]
 XWD SSNPC%,[ASCIZ/No CI port on specified CPU/]
 XWD SSNNK%,[ASCIZ/CPU's node number not known/]
 XWD SSINN%,[ASCIZ/Invalid CI node number/]
 XWD SSNFC%,[ASCIZ/No free core/]
 XWD SSVNO%,[ASCIZ/Virtual circuit not open/]
 XWD SSICI%,[ASCIZ/Invalid connect id/]
 XWD SSRQE%,[ASCIZ/Receive queue empty/]
 XWD SSNBQ%,[ASCIZ/No buffer queued for packet reception/]
 XWD SSRCF%,[ASCIZ/Reject connection failed/]
 XWD SSDCF%,[ASCIZ/Disconnect connection failed/]
 XWD SSNFB%,[ASCIZ/No free buffers to send packet/]
 XWD SSQBF%,[ASCIZ/Queue buffers failed/]
 XWD SSCBF%,[ASCIZ/Cancel buffers failed/]
 XWD SSPSF%,[ASCIZ/Packet send failed/]
 XWD SSDQE%,[ASCIZ/Data entry queue empty/]
 XWD SSEQE%,[ASCIZ/Event queue empty/]
 XWD SSCRB%,[ASCIZ/Can't remove buffer from database/]
 XWD SSCUB%,[ASCIZ/Can't unmap buffer/]
 XWD SSNSB%,[ASCIZ/No such buffer name/]
 XWD SSTMS%,[ASCIZ/Too many buffer segment descriptors/]
 XWD SSIDM%,[ASCIZ/Illegal data mode/]
 XWD SSSCP%,[ASCIZ/Segment crosses page boundary/]
 XWD SSSLT%,[ASCIZ/Segment too long (greater than 1 page)/]
 XWD 777777,[ASCIZ/Unknown error code/]
 ERRSCL==.-ERRSCT-1 ;Number of errors-1

DFCIB1 CTP Connect/Disconnect/Status Subroutines version 2(20) MACRO %53B(1242) 17:32 27-Aug-85 Page 32
DFCIB1 MAC 27-Aug-85 11:17 Literals

SEQ 0175

```
1098 SUBTTL Literals
1099
1100 006171' LIT..1: XLIST
1101 LIST
1102
1103 END
```

NO ERRORS DETECTED

PROGRAM BREAK IS 010041
CPU TIME USED 01:23.801

158P CORE USED

BUFLNM	80#																		
BUFRNM	80#	824	847																
CHKEV	76#	709																	
CHKST	76#	664	683																
CONID	67	106#	398	430	445	483	504	626											
CONN	223	258	427#																
CONNB	71	262	427	428	432	434	436	438	441	444	448#								
CONND	68	453	464#																
COP	460#	464																	
CREV	462#	464																	
CVER	461#	464																	
CWAIT	68	651#																	
CWAIT1	671#	679																	
CWAIT2	667	683#																	
CWAIT3	686#	694																	
CWAIT7	678	693	696#																
CWAIT8	676	691	697#																
CYCL60	731	737	746	749#															
DBUF1	69	96#	210	222	245	257													
DBUF2	69	97#	209	212	244	247													
DBUF3	69	98#	211	214	246	249													
DBUF4	70	99#	213	216	248	251													
DBUF5	70	100#	215	218	250	253													
DBUF6	70	101#	217	219	252	254													
DECV	59																		
DIAG2	900#	907																	
DIAGAG	909#	938	941	942	943	945													
DIAGJ	890#	947																	
DIAGU	892	903#																	
DISC	67	482#																	
DISCB	484	486	489	493#															
DISCR	107#	490																	
DOCON	67	195#																	
DOCONX	72	230#																	
DODIS	67	481#																	
ERRSCL	1052	1097#																	
ERRSCT	1053	1056	1067#	1097															
EVENTB	66	586	627	629	630	631	632	633	634	635	638	644#	713						
EWAIT	68	708#																	
EWAIT7	712	718#																	
EWAIT8	716	720#																	
FLGFLG	84#	813																	
GETTI1	726	730#																	
GETTI2	732	740#																	
GETTIM	725#	759	777	787	791														
GEVNT	66	579	625#	710															
GLNN	67	371#																	
GPRS	72	977#																	
GPRSB	72	977	979	982	989#														
LIT..1	1100#																		
LNODB	381	384	388#																
LNODE	72	371	379#																
LNODEN	72	105#	385																

SEQ 0176

[illegible]

S1

130	136	139	142	145	148	153	154	156	158	160	181	195	196
197	198	199	200	201	202	203	204	205	209	210	211	212	213
214	215	216	217	218	221	230	231	232	233	234	235	236	237
238	239	240	244	245	246	247	248	249	250	251	252	253	256
261	264	265	267	275	279	280	284	286	288	290	295	296	298
300	302	304	306	308	310	312	314	316	318	329	333	334	336
338	340	342	344	346	348	350	352	354	356	379	380	384	385
397	398	399	400	401	402	427	429	431	432	433	434	435	436
437	438	440	444	445	482	483	484	485	489	490	503	504	505
506	528	529	532	533	534	577	586	587	589	591	593	595	597
599	601	603	605	607	609	611	625	626	627	628	629	637	661
664	665	666	675	683	684	689	690	708	709	715	733	734	735
736	737	740	741	742	743	744	745	746	758	760	761	762	776
778	785	786	788	792	802	804	805	806	807	808	809	815	816
817	819	823	824	846	847	848	849	864	865	869	870	871	872
890	903	935	937	938	939	941	945	977	978	981	1001	1015	1023

S2

1024	1025	1026	1029	1030	1032	1033	1035	1036	1053	1054			
129	138	141	144	147	150	152	155	157	159	161	182	183	195
222	230	257	262	265	266	267	268	275	280	281	282	294	297
299	301	303	305	307	309	311	313	315	317	319	329	332	335
337	339	341	343	345	347	349	351	353	355	357	379	381	397
403	428	429	441	482	486	503	507	527	529	535	577	585	588
590	592	594	596	598	600	602	604	606	608	610	612	625	638
661	673	674	675	688	689	690	708	713	714	715	730	758	776
786	802	820	846	850	864	866	890	896	897	935	978	982	1002

SCOUNT

SCS\$

SCS\$1

SCS\$2

SCSBLK

SCSCAL

SCSER1

SCSER2

SCSER3

SCSER4

SCSERC

SCSERR

SCSERS

SCSFCN

SECS

SEVNT

SHCST

SHPOLL

SQZACS

SQZCAK

SQZCLO

SQZCRE

SQZCSE

SQZCVD

SQZDAK

SQZDMC

SQZDRE

SEQ 0178

[illegible]

WAITXE
\$RETF
\$RETIF
\$RETI
\$RETI

..0001
..0002
..0003
..0004
..0005
..0006
..0007
..0010
..0011
..0012
..0013
..0014
..0015
..0016
..0017
..0020
..0021
..0022
..0023
..0024
..0025
..0026
..0027
..0030
..0031
..0032

..MX1
..MX2
..TSA1

.A13

.A14

.A15

.A16

.ACB

110# 785 788 792

63
63
63
63
127#
181#
196#
231#
276#
330#
380#
398#
430#
483#
504#
530#
578#
626#
662#
709#
731#
759#
777#
787#
803#
847#
865#
891#
936#
979#
1035#
1035#
1035#
126#
397#
661#
846#
127#
661#
127#
661#
127#
661#
126#
625#
126#
276#
429#
530#
708#
777#
864#
979#

196#
231#
276#
330#
380#
398#
430#
483#
504#
530#
578#
626#
662#
709#
731#
759#
777#
787#
803#
847#
865#
891#
936#
979#
1035#
1036#

1036

180#
429#
708#
864#
181#
195#
730#
758#
181#
195#
730#
758#
181#
195#
730#
758#
180#
180#
329#
329#
430#
430#
577#
577#
709#
709#
786#
786#
865#
865#

195#
482#
730#
890#
275#
329#
776#
786#
275#
329#
776#
786#
275#
329#
776#
786#
195#
195#
330#
330#
482#
482#
578#
578#
730#
730#
787#
787#
890#
890#

230#
503#
758#
935#
379#
397#
802#
846#
379#
397#
802#
846#
379#
397#
802#
846#
196#
196#
379#
379#
483#
483#
625#
625#
731#
731#
802#
802#
891#
891#

275#
529#
776#
978#
429#
482#
864#
890#
429#
482#
864#
890#
397#
429#
846#
864#
230#
230#
380#
380#
503#
503#
626#
626#
758#
758#
803#
803#
935#
935#

329#
577#
786#
978#
503#
529#
935#
978#
503#
529#
935#
978#
503#
529#
935#
978#
231#
231#
397#
397#
504#
504#
661#
661#
759#
759#
846#
846#
936#
936#

329#
577#
786#
978#
529#
577#
978#
978#
529#
577#
978#
978#
529#
577#
978#
978#
275#
275#
398#
398#
529#
529#
662#
662#
776#
776#
847#
847#
978#
978#

SEQ 0180

[illegible]

.SQDSV	136				
.SQDTA	632	633	634	635	
.SQESB	630				
.SQEVT	586	631	713		
.SQFST	279	295	673	688	1036
.SQLDG	871				
.SQLEN	432	533	629		
.SQLMG	869				
.SQLNN	384				
.SQNAM	848				
.SQNNM	142	145			
.SQOSB	528				
.SQRCI	444				
.SQRPN	977				
.SQRPS	979	989			
.SQSPN	436				
.SQSYS	434				
.SQVCS	153				
.SQXFL	809	815	817		
.SSCON	440				
.SSCSP	402				
.SSDIS	485				
.SSEVT	637				
.SSGLN	380				
.SSMAP	819				
.SSRBS	865				
.SSRCD	534				
.SSRPS	981				
.SSSTS	506	1033			
.SSUMP	849				

SEQ 0182

.NV4

891#

891

SEQ 0184


```
1      ;Z:<GSCOTT.DFCIB>DFCIB2.MAC.56 21-Aug-85 17:45:12, Edit by GSCOTT
2      ;Use symbols in SND1.
3      ;Z:<GSCOTT.DFCIB>DFCIB2.MAC.55 21-Aug-85 16:28:00, Edit by GSCOTT
4      ;Remove repeat 0ed code
5      ;Z:<GSCOTT.DFCIB>DFCIB2.MAC.53 14-Aug-85 15:57:38, Edit by GSCOTT
6      ;Replace inline SCS JSYS calls with SCS$.
7      ;Z:<GSCOTT.DFCIB>DFCIB2.MAC.50 9-Aug-85 10:26:22, Edit by GSCOTT
8      ;Calling RCON in MOVBUF was causing a bug and wasn't needed anyway, so zap it.
9      ;Z:<GSCOTT.DFCIB>DFCIB2.MAC.38 6-Aug-85 19:36:24, Edit by GSCOTT
10     ;Use $SAVE for all AC save/rest
11     ;Z:<GSCOTT.DFCIB>DFCIB2.MAC.31 6-Aug-85 19:18:46, Edit by GSCOTT
12     ;Remove routines not called from anywhere: MSGWT, STATS, SEVENT, CLRSAV, CPYBFR
13     ;RSPSAV, CPYSCS. Some of them were riddled with bugs anyway.
14     ;Z:<GSCOTT.DFCIB>DFCIB2.MAC.30 6-Aug-85 19:08:58, Edit by GSCOTT
15     ;Add new title statements, change AC names, begin $TEXTing, etc.
16     ;Z:<GSCOTT.DFCIB>DFCIB2.MAC.22 5-Aug-85 13:36:14, Edit by GSCOTT
17     ;Begin GLXLIB conversion: RCON arguments in S1, S2.
```

```
18 SEARCH DFCIBT
19 SALL
20 .DIREC FLBLST
21 NOSYM
22 NAME (DFCIB2,\DECVER,\VEDIT,CTP Packet Build/Send/Receive Subroutines)
23
24 SEARCH GLXMAC,MONSYM,UUOSYM,MACSYM
25 PROLOG (DFCIB2)
26
27
```

28 ;Interns

```
29
30 INTERN BLDCTP,SNDMSG,OPCODE,GENLEN,READM,EXAMNM,REQUEM
31 INTERN BUFTYP,RCOUNT,GENFUN,GNCNST,BUFLEN,BUFLNM,DELAY
32 INTERN MOVTYP,PKTSIZ,OTHNOD,PKTMLT,LOFSET,BUFRNM,ROFSET
33 INTERN GENLEN,IMGDAT,ABUFNM,ANMREQ,RETBUF,CTPLEN,RLENBF
34 INTERN CTPPKT,SCSCMD,SCSRSP,SCSEVT,SAVREQ,SAVCTP,SAVSCS,SCSSAV
35 INTERN SAVRSP,SNDDAT,READD,REQUED,EXTEND,ACTFLG,NUMREF
36 INTERN BFLNM1,DMARD,DMADRS,STADR
37
```

38 ;Externs in DFCIB1

```
39 EXTERN SCSS$,SCSERR,CONID,RCON,CONNB,RCONB
40
```

41 ;Externs in DFCIBP

```
42 EXTERN PATHS
43
```

44 ;Externs in DFCIBM

```
45 EXTERN HSCDST,T20DST,VMSDST,T10DST,ITERCT,IMAGDT,IMGLEN
46
47
48
49
```

SUBTTL Footprint Definitions

;CTP\$OPCODE DEFINITIONS USED BY THE CONTROLLER

50			
51			
52			
53			
54	000000	CTPFST==0	: FUNCTION SET
55	000001	CTPMAP==1	: BUFFER MAP REQUEST
56	000002	CTPUNM==2	: UNMAP A BUFFER
57	000003	CTPMOV==3	: MOVE A BUFFER REQUEST
58	000004	CTPMSG==4	: GENERATE A MESSAGE REQUEST
59	000005	CTPDGM==5	: GENERATE A DATAGRAM REQUEST
60	000011	CTPCFG==11	: CONFIGURATION REQUEST
61	000012	CTPCNT==12	: COUNTER READ REQUEST
62	000013	CTPCON==13	: 3RD PARTY CONNECT
63	000014	CTPDSC==14	: RESPONDER DISCONNECT
64	000015	CTPACT==15	: GENERATE ACTIVITY REQUEST

```

65          SUBTTL  Build CTP Packets
66
67          ;THIS ROUTINE WILL CONSTRUCT A CTP PACKET FOR THE DESIRED
68          ;CTP FUNCTION.  S1 WILL CONTAIN THE DESIRED FUNCTION.
69
70          BLDCTP: $SAVE  <S1,S2,T1>          ; SAVE AC'S
71          MOVE  S1,OPCODE          ; OPCODE TO S1
72          CAIN  S1,CTPFST          ; TEST IF FUNCTION SET
73          JRST  FSÉT              ; YES, CREATE PACKET
74          CAIN  S1,CTPMAP          ; TEST IF BUFFER MAP REQUEST.
75          JRST  BMAP              ; YES, CREATE PACKET
76          CAIN  S1,CTPUNM          ; TEST IF BUFFER UNMAP REQUEST.
77          JRST  BUNMAP            ; YES, CREATE PACKET.
78          CAIN  S1,CTPMOV          ; TEST IF MOVE BUFFER REQUEST.
79          JRST  MOVBUF            ; YES, CREATE PACKET.
80          CAIN  S1,CTPMMSG          ; TEST IF GENERATE MSG.
81          JRST  GENMSG            ; YES, CREATE PACKET
82          CAIN  S1,CTPDGM          ; TEST IF GENERATE DATAGRAM.
83          JRST  GENDGM            ; YES, CREATE PACKET.
84          CAIN  S1,CTPCFG          ; TEST IF CONFIGURATION DATA.
85          JRST  CONFIG            ; YES, CREATE PACKET.
86          CAIN  S1,CTPCNT          ; TEST IF READ COUNTER.
87          JRST  CNTRD             ; YES, CREATE PACKET.
88          CAIN  S1,CTPCON          ; TEST IF CONNECT
89          JRST  CONNCT            ; YES, CREATE PACKET.
90          CAIN  S1,CTPDSC          ; TEST IF LISTENER DISCONNECT.
91          JRST  LDISC             ; YES, CREATE PACKET.
92          CAIN  S1,CTPACT          ; TEST IF GENERATE ACTIVITY.
93          JRST  GENACT            ; YES, CREATE PACKET.
94
95          000035' 26.1 17 0 00 000000*  BLDERR: $TEXT  (,<? Unsupported function ^D/S1/. in BLDCTP>)
96

```

```

97
98
99
100 000037' 304 00 0 00 000000
101 000045' 260 17 0 00 001252'
102 000046' 201 02 0 00 001321'
103 000047' 137 01 0 00 001271'
104 000050' 260 17 0 00 000644'
105 000051' 260 17 0 00 000631'
106 000052' 260 17 0 00 001262'
107 000053' 201 01 0 00 000005'
108 000054' 202 01 0 00 002216'
109 000055' 263 17 0 00 000000

;Function set
FSET:  $SAVE  <S1,S2,T1>      ; SAVE AC'S
        $CALL  CLRCTP         ; CLEAR CTPPKT AREA.
        MOVEI  S2,CTPPKT
        DPB    S1,CTPOP       ; OP CODE TO CTPPKT.
        $CALL  NXTREF         ; GET NEXT REF NUMBER.
        $CALL  REFNUM         ; STORE IN CTP
        $CALL  REQSAV         ; SAVE CTP REQUEST.
        MOVEI  S1,5           ; SAVE LENGTH OF CTP
        MOVEM  S1,CTPLEN      ; FOR SCSCMD PACKET.
        $RET
    
```



```

110
111      ;Buffer map
112
113      000056' 304 00 0 00 000000      BMAP:  $SAVE  <S1,S2,T1>      ; SAVE AC'S
114      000064' 260 17 0 00 001252'      $CALL  CLRCTP      ; CLEAR CTPPKT AREA.
115      000065' 201 02 0 00 001321'      MOVEI   S2,CTPPKT
116      000066' 137 01 0 00 001271'      DPB      S1,CTPOP      ; OP CODE TO CTPPKT.
117      000067' 260 17 0 00 000644'      $CALL  NXTREF      ; GET NEXT REF NUMBER.
118      000070' 260 17 0 00 000631'      $CALL  REFNUM      ; STORE IN CTP
119      000071' 200 01 0 00 002247'      MOVE    S1,BUFTYP      ; GET BUFTYP FROM PARSER
120      000072' 137 01 0 00 001273'      DPB      S1,BFTYP1      ; BUFTYP TO S1.
121
122      ;TEST BUFTYP = 1 TO 2.
123
124      000073' 305 01 0 00 000001      CAIGE    S1,1      ; GENFUN >= 0.
125      000074' 254 00 0 00 000136'      JRST    BTYPER      ; NO, GO TO PRINT ERROR.
126      000075' 303 01 0 00 000002      CAILE    S1,2      ; GENFUN <= 2.
127      000076' 254 00 0 00 000136'      JRST    BTYPER      ; NO, GO TO PRINT ERROR.
128      000077' 137 01 0 00 001273'      DPB      S1,BFTYP1      ; BUFTYP TO BUFMAP CTPPKT.
129
130      ;SET ZEROS IN TWO BYTES
131
132      000100' 400 01 0 00 000000      SETZ     S1,
133      000101' 137 01 0 00 001274'      DPB      S1,BMAPA
134
135      000102' 200 01 0 00 002232'      MOVE     S1,GENFUN      ; GENFUN TO S1
136
137      ;TEST GENFUN = 0 TO 2.
138
139      000103' 305 01 0 00 000000      CAIGE    S1,0      ; GENFUN >= 0.
140      000104' 254 00 0 00 000133'      JRST    GNFERR      ; NO, GO TO PRINT ERROR.
141      000105' 303 01 0 00 000002      CAILE    S1,2      ; GENFUN <= 2.
142      000106' 254 00 0 00 000133'      JRST    GNFERR      ; NO, GO TO PRINT ERROR.
143      000107' 137 01 0 00 001275'      DPB      S1,GNFNC1      ; GENFUN TO BUFMAP CTPPKT.
144
145      ;IF CTP$GENFUNCT=CTP$GENFILL,THEN GENCONST TO CTPPKT.
146
147      000110' 302 01 0 00 000000      CAIE     S1,0      ; TEST IF GENFUNCT=GENFILL.
148      000111' 254 00 0 00 000114'      JRST    BMAP1      ; NO, CONTINUE W/O GENCONST.
149      000112' 200 01 0 00 002233'      MOVE     S1,GNCNST
150      000113' 137 01 0 00 001276'      DPB      S1,GCNST1      ; YES, LOAD CTPPKT WITH CONSTANT.

```

```

151
152
153      ;SET ZEROS IN TWO BYTES
154
155
156      ;TEST PKTSIZ = 0 TO 1.
157
158      000114' 200 01 0 00 002225'  BMAP1: MOVE      S1,PKTSIZ
159      000115' 305 01 0 00 000000'    CAIGE      S1,0          ; GENFUN >= 0.
160      000116' 254 00 0 00 000271'    JRST      PSZER1      ; NO, GO TO PRINT ERROR.
161      000117' 303 01 0 00 000001'    CAILE      S1,1          ; GENFUN <= 1.
162      000120' 254 00 0 00 000271'    JRST      PSZER1      ; NO, GO TO PRINT ERROR.
163      000121' 137 01 0 00 001302'    DPB        S1,PKTSZ1    ; STORE PACKET SIZE INTO CTPPKT.
164
165      000122' 200 01 0 00 002227'    MOVE      S1,PKTMLT
166      000123' 137 01 0 00 001304'    DPB        S1,PKMLT1    ; STORE MOVETYPE IN CTPPKT.
167
168      000124' 200 01 0 00 002212'    MOVE      S1,BUFLEN
169      000125' 260 17 0 00 001222'    $CALL     RVSNUM
170      000126' 202 01 0 02 000003'    MOVEM     S1,3(S2)
171
172      000127' 201 01 0 00 000020'    BMAPEX: MOVEI    S1,20          ; SAVE LENGTH OF CTP
173      000130' 202 01 0 00 002216'    MOVEM     S1,CTPLEN      ; FOR SCSCMD PACKET.
174      000131' 260 17 0 00 001262'    $CALL     REQSAV      ; SAVE CTP REQUEST
175      000132' 263 17 0 00 000000'    $RET
176
177      ;ERROR PRINTOUT ROUTINES
178
179      000133' 260 17 0 00 000035*  GNFERR: $TEXT    (,<? GENFUNCTION not valid on buffer map command>)
180      000135' 254 00 0 00 000127'    JRST      BMAPEX
181
182      000136' 260 17 0 00 000133*  BTYPER: $TEXT    (,<? BUFTYPE not valid on buffer map command>)
183      000140' 254 00 0 00 000127'    JRST      BMAPEX
  
```

```

184
185
186      ;Buffer unmap
187 000141' 304 00 0 00 000000      BUNMAP: $SAVE    <S1,S2,T1>      ; SAVE AC'S
188 000147' 260 17 0 00 001252'      $CALL    CLRCTP      ; CLEAR CTPPKT AREA.
189 000150' 201 02 0 00 001321'      MOVEI    S2,CTPPKT
190 000151' 137 01 0 00 001271'      DPB      S1,CTPOP      ; OP CODE TO CTPPKT.
191 000152' 260 17 0 00 000644'      $CALL    NXTREF      ; GET NEXT REF NUMBER.
192 000153' 260 17 0 00 000631'      $CALL    REFNUM      ; STORE IN CTP
193 000154' 200 01 0 00 002247'      MOVE     S1,BUFTYP      ; GET BUFTYP FROM PARSER.
194
195      ;TEST BUFTYP = 0 TO 2.
196
197 000155' 305 01 0 00 000000      CAIGE    S1,0      ; GENFUN >= 0.
198 000156' 254 00 0 00 000176'      JRST    BTPER1      ; NO, GO TO PRINT ERROR.
199 000157' 303 01 0 00 000002'      CAILE    S1,2      ; GENFUN <= 2.
200 000160' 254 00 0 00 000176'      JRST    BTPER1      ; NO, GO TO PRINT ERROR.
201 000161' 137 01 0 00 001273'      DPB      S1,BFTYP1      ; BUFTYP TO BUFPAP CTPPKT.
202
203
204      ;SET ZEROS IN BYTES 6 THRU' 11.
205
206 000162' 400 00 0 00 000001'      SETZ    S1
207 000163' 137 01 0 00 001274'      DPB      S1,BMAPA
208 000164' 402 00 0 02 000002'      SETZM    2(S2)
209
210 000165' 200 01 0 00 002212'      MOVE     S1,BUFLN
211 000166' 260 17 0 00 001222'      $CALL    RVSNUM
212 000167' 202 01 0 02 000003'      MOVEM   S1,3(S2)
213
214      ;DONT KNOW IF LEFT SHIFT NEEDED-MAY BE NORMALIZED OFF THE WIRE.
215
216 000170' 200 01 0 00 002213'      MOVE     S1,BUFLNM
217 000171' 202 01 0 02 000004'      MOVEM   S1,4(S2)
218
219 000172' 201 01 0 00 000024'      UNMPEX: MOVEI    S1,24      ; SAVE LENGTH OF CTP
220 000173' 202 01 0 00 002216'      MOVEM   S1,CTPLEN      ; FOR SCSCMD PACKET.
221 000174' 260 17 0 00 001262'      $CALL    REQSAV      ; SAVE CTP REQUEST
222 000175' 263 17 0 00 000000      $RET
223
224      ;ERROR PRINTOUT ROUTINES
225
226
227 000176' 260 17 0 00 000136*      BTPER1: $TEXT    (<? BUFTYPE not valid on buffer unmap command>)
228 000200' 254 00 0 00 000172'      JRST    UNMPEX

```

```

229
230      ;Move buffer
231
232      000201' 304 00 0 00 000000      MOVBUF: $SAVE    <S1,S2,T1>      ; SAVE AC'S
233      000207' 260 17 0 00 001252'      $CALL    CLRCTP      ; CLEAR CTPPKT AREA.
234      000210' 201 02 0 00 001321'      MOVEI    S2,CTPPKT
235      000211' 137 01 0 00 001271'      DPB      S1,CTPOP      ; OP CODE TO CTPPKT.
236      000212' 260 17 0 00 000644'      $CALL    NXTREF      ; GET NEXT REF NUMBER.
237      000213' 260 17 0 00 000631'      $CALL    REFNUM      ; STORE REF NUMBER IN CTP
238      000214' 200 01 0 00 002234'      MOVE     S1,DELAY
239      000215' 137 01 0 00 001300'      DPB      S1,DELAY1      ; STORE DELAY IN CTPPKT.
240
241
242      ;STORE REPEAT COUNT IN CTPPKT.
243
244      000216' 200 01 0 00 002235'      MOVE     S1,RCOUNT
245      000217' 137 01 0 00 001305'      DPB      S1,RCNT1
246      000220' 242 01 0 00 777770'      LSH      S1,-8
247      000221' 137 01 0 00 001306'      DPB      S1,RCNT2
248      000222' 200 01 0 00 002221'      MOVE     S1,MOVTYP
249
250      ;TEST MOVTYP = 0 TO 3.
251
252      000223' 305 01 0 00 000000'      CAIGE    S1,0      ; GENFUN >= 0.
253      000224' 254 00 0 00 000277'      JRST     MTPER1      ; NO, GO TO PRINT ERROR.
254      000225' 303 01 0 00 000003'      CAILE    S1,3      ; GENFUN <= 2.
255      000226' 254 00 0 00 000277'      JRST     MTPER1      ; NO, GO TO PRINT ERROR.
256      000227' 137 01 0 00 001301'      DPB      S1,MVTYP1      ; STORE MOVETYPE IN CTPPKT.
257
258      000230' 200 01 0 00 002225'      MOVE     S1,PKTSIZ
259
260      ;TEST PKTSIZ = 0 TO 1.
261
262      000231' 305 01 0 00 000000'      CAIGE    S1,0      ; GENFUN >= 0.
263      000232' 254 00 0 00 000274'      JRST     PSZER3      ; NO, GO TO PRINT ERROR.
264      000233' 303 01 0 00 000001'      CAILE    S1,1      ; GENFUN <= 1.
265      000234' 254 00 0 00 000274'      JRST     PSZER3      ; NO, GO TO PRINT ERROR.
266      000235' 137 01 0 00 001302'      DPB      S1,PKTSZ1      ; STORE PACKET SIZE INTO CTPPKT.
267
268      000236' 200 01 0 00 002226'      MOVE     S1,OTHNOD
269      000237' 137 01 0 00 001303'      DPB      S1,OTHND1      ; STORE OTHERNODE IN CTPPKT.
270
271      ;TEST PKTMULT = 0 TO 1.
272
273      000240' 200 01 0 00 002227'      MOVE     S1,PKTMLT
274      000241' 137 01 0 00 001304'      DPB      S1,PKMLT1      ; STORE MOVETYPE IN CTPPKT.

```

```

275
276      ;REVERSE BUFLNGTH AND STORE IN CTPPKT.
277
278      000242' 200 01 0 00 002212'      MOVE      S1,BUFLN      ; GET BUF LENGTH
279      000243' 260 17 0 00 001222'      $CALL     RV$NUM
280      000244' 202 01 0 02 000003'      MOVEM     S1,3(S2)
281
282      ;STORE LOCAL BUFFER NAME
283
284      000245' 200 01 0 00 002213'      MOVE      S1,BUFLNM
285      000246' 202 01 0 02 000004'      MOVEM     S1,4(S2)
286
287      ;STORE LOCAL OFFSET
288
289      000247' 200 01 0 00 002214'      MOVE      S1,LOFSET
290      000250' 260 17 0 00 001222'      $CALL     RV$NUM
291      000251' 202 01 0 02 000005'      MOVEM     S1,5(S2)
292
293      000252' 200 01 0 00 002231'      MOVBF1: MOVE     S1,BUFRNM
294      000253' 242 01 0 00 777774'      LSH        S1,-4
295      000254' 260 17 0 00 001222'      $CALL     RV$NUM
296      000255' 202 01 0 02 000006'      MOVEM     S1,6(S2)
297      000256' 254 00 0 00 000262'      JRST      MOVBF3
298
299      000257' 200 01 0 00 002231'      MOVBF2: MOVE     S1,BUFRNM
300      000260' 242 01 0 00 777774'      LSH        S1,-4
301      000261' 202 01 0 02 000006'      MOVEM     S1,6(S2)
302
303      000262' 200 01 0 00 002230'      MOVBF3: MOVE     S1,ROFSET
304      000263' 260 17 0 00 001222'      $CALL     RV$NUM
305      000264' 202 01 0 02 000007'      MOVEM     S1,7(S2)
306
307      000265' 201 01 0 00 000040'      MVBFE1: MOVEI     S1,40      ; SAVE LENGTH OF CTP
308      000266' 202 01 0 00 002216'      MOVEM     S1,CTPLEN      ; FOR SCSCMD PACKET.
309      000267' 260 17 0 00 001262'      $CALL     REQSAV      ; SAVE CTP REQUEST
310      000270' 263 17 0 00 000000'      $RET
311
312      ;Error printout routines
313
314      000271' 260 17 0 00 000176*      PSZER1: $TEXT      (,<? Packet size not valid on map buffer command>)
315      000273' 254 00 0 00 000127'      JRST      BMAPEX
316
317      000274' 260 17 0 00 000271*      PSZER3: $TEXT      (,<? Packet size not valid on buffer move command>)
318      000276' 254 00 0 00 000265'      JRST      MVBFE1
319
320      000277' 260 17 0 00 000274*      MTPER1: $TEXT      (,<? Move type not valid on buffer move command>)
321      000301' 254 00 0 00 000265'      JRST      MVBFE1

```



```

322
323
324
325 000302' 304 00 0 00 000000'
326 000310' 260 17 0 00 001252'
327 000311' 201 02 0 00 001321'
328 000312' 137 01 0 00 001271'
329 000313' 260 17 0 00 000644'
330 000314' 260 17 0 00 000631'
331
332 000315' 200 01 0 00 002234'
333 000316' 137 01 0 00 001300'
334
335
336
337 000317' 200 01 0 00 002235'
338 000320' 137 01 0 00 001305'
339 000321' 242 01 0 00 777770'
340 000322' 137 01 0 00 001306'
341
342 000323' 200 01 0 00 002232'
343
344
345
346 000324' 305 01 0 00 000000'
347 000325' 254 00 0 00 000402'
348 000326' 303 01 0 00 000003'
349 000327' 254 00 0 00 000402'
350 000330' 137 01 0 00 001275'
351
352
353
354 000331' 302 01 0 00 000000'
355 000332' 254 00 0 00 000335'
356 000333' 200 01 0 00 002273'
357 000334' 137 01 0 00 001276'

;Generate message
GENMSG: $SAVE <S1,S2,T1>
        $CALL CLRCTP
        MOVEI S2,CTPPKT
        DPB S1,CTPOP
        $CALL NXTREF
        $CALL REFNUM
        MOVE S1,DELAY
        DPB S1,DELAY1
        ; SAVE AC'S
        ; CLEAR CTPPKT AREA.
        ; OP CODE TO CTPPKT.
        ; GET NEXT REF NUMBER.
        ; STORE REF NUMBER IN CTP
        ; STORE DELAY IN CTPPKT.

;STORE REPEAT COUNT IN CTPPKT.
        MOVE S1,RCOUNT
        DPB S1,RCNT1
        LSH S1,-8
        DPB S1,RCNT2
        MOVE S1,GENFUN
        ; GENFUN TO S1.

;TEST GENFUN = 0 TO 3.
        CAIGE S1,0
        JRST GNMGER
        CAILE S1,3
        JRST GNMGER
        DPB S1,GNFNC1
        ; GENFUN >= 0.
        ; NO, GO TO PRINT ERROR.
        ; GENFUN <= 3.
        ; NO, GO TO PRINT ERROR.
        ; GENFUN TO GENMSG CTPPKT.

;IF CTP$GENFUNCT=CTP$GENFILL, THEN GENCONST TO CTPPKT.
        CAIE S1,0
        JRST GNMMSG1
        MOVE S1,GCNST
        DPB S1,GCNST1
        ; TEST IF GENFUNCT=GENFILL.
        ; NO, CONTINUE W/O GENCONST.
        ; YES, LOAD CTPPKT WITH CONSTANT.

```

```

358
359 ;SET ZEROS IN TWO BYTES
360
361 000335' 400 01 0 00 00 000' GNMSG1: SETZ S1,
362 000336' 137 01 0 00 00 277' DPB S1,BMAPB
363
364 ;STORE DELAY COUNT IN CTPPKT.
365
366 000337' 200 01 0 00 00 002237' MOVE S1,GENLEN
367 000340' 137 01 0 00 00 001307' DPR S1,GNLEN1 ; STORE DELAY IN CTPPKT.
368 000341' 242 01 0 00 777770' LSH S1,-8 ; SHIFT M/S BYTE TO L/S BYTE
369 000342' 137 01 0 00 001310' DPB S1,GNLEN2 ; STORE DELAY IN CTPPKT.
370
371 ;BUILD IMAGE BUFFER (LIMITED TO 4 BYTES).
372 ;DATA LOADED INTO 'IMAGDT'.
373
374 000343' 200 01 0 00 00 002232' MOVE S1,GENFUN ; TEST IF GENFUNCT=GENFIMAGE
375 000344' 302 01 0 00 00 000003' CAIE S1,3
376 000345' 254 00 0 00 00 000376' JRST GNMGEX ; GENFUNCT NOT EQUAL GENFIMAGE,SKIP.
377
378 ;
379 ; data generation routine for image data.
380 ;
381 ; +-----+-----+-----+-----+
382 ; ! byte0 ! byte1 ! byte2 ! byte3 !XXX!
383 ; +-----+-----+-----+-----+
384 ;
385 ; register usage for ctppkt data:
386 ; S1=ADDRESS OF CTPPKT s2=pointer to ctppkt byte
387 ;
388 ; register usage for IMAGE DATA:
389 ; t2=byte holding register t3=address of imagdt
390 ; t4=pointer to image data byte p1=image data buffer length
391
392 000346' 201 01 0 00 00 001321' MOVEI S1,CTPPKT
393 000347' 200 02 0 00 00 002420' MOVE S2,[POINT 8,3(S1),15] ; BYTE POINTER TO CTPPKT BYTE
394 000350' 201 03 0 00 00 000000' MOVEI T1,0 ; RESET
395 000351' 201 05 0 00 00 000000* MOVEI T3,IMAGDT
396 000352' 275 05 0 00 00 000001' SUBI T3,1
397 000353' 200 06 0 00 00 002421' MOVE T4,[POINT 8,(T3),31] ; BYTE POINTER TO IMAGE DATA BYTE
398 000354' 201 07 0 00 00 000000' MOVEI P1,0

```

```

396
397
398
399 000355' 134 04 0 00 000006
400 000356' 136 04 0 00 000002
401 000357' 271 07 0 00 000001
402 000360' 271 03 0 00 000001
403 000361' 312 03 0 00 000000*
404 000362' 254 00 0 00 000364'
405 000363' 254 00 0 00 000371'
406
407 000364' 312 07 0 00 000000*
408 000365' 254 00 0 00 000355'
409 000366' 201 07 0 00 000000
410 000367' 200 06 0 00 002421'
411 000370' 254 00 0 00 000355'
412
413
414
415
416 000371' 201 07 0 00 000000
417
418
419
420 000372' 201 01 0 00 000016
421 000373' 270 01 0 00 000361*
422 000374' 202 01 0 00 002216'
423 000375' 254 00 0 00 000400'
424
425 000376' 201 01 0 00 000020
426 000377' 202 01 0 00 002216'
427 000400' 260 17 0 00 001262'
428 000401' 263 17 0 00 000000
429
430
431
432 000402' 260 17 0 00 000277*
433 000404' 254 00 0 00 000376'

;GET IMAGE DATA BYTE AND PUT IN CTPPKT.
GNMSG2: ILDB T2,T4 ; T2= DATA HOLDING REG(FROM IMAGDT)
        IDPB T2,S2 ; T2 TO IMAGE DATA IN CTPPKT.
        ADDI P1,1 ; INCR LOCAL IMAGE COUNTER.
        ADDI T1,1 ; INCR LOCAL TOTAL COUNTER
        CAME T1,ITERCT ; TEST IF TOTAL COUNT = ITERCT.
        JRST GNMSG3 ; CONTINUE TO STUFF BUFFER.
        JRST GNMSG5 ; EXIT, BUFFER STUFFING COMPLETED.

GNMSG3: CAME P1,IMGLN
        JRST GNMSG2
        MOVEI P1,0 ; LOCAL IMAGE CNTR = 0.
        MOVE T4,[POINT 8,(T3),31] ; BYTE POINTER TO IMAGE DATA BYTE
        JRST GNMSG2 ; LOOP BACK TO WRITE MORE IMAGE DATA.

;RESET POINTER TO BEGINNING OF IMAGE DATA.
GNMSG5: MOVEI P1,0 ; RESET IMAGE DATA COUNTER

;ADJUST LENGTH OF CTPPKT.
        MOVEI S1,16 ; LENGTH OF BASE PACKET
        ADD S1,ITERCT ; ADD LENGTH OF IMAGE DATA.
        MOVEM S1,CTPLEN ; STORE IN CTPPKT LENGTH.
        JRST GNMGE1

GNMGEX: MOVEI S1,20 ; SAVE LENGTH OF CTP
        MOVEM S1,CTPLEN ; FOR SCSCMD PACKET.
GNMGE1: $CALL REQSAV ; SAVE CTP REQUEST
        $RET

;ERROR PRINTOUT ROUTINES
GNMGER: $TEXT (,<? Generate function not valid on generate message command>)
        JRST GNMGEX

```

```

434
435
436
437 000405' 304 00 0 00 000000'
438 000413' 260 17 0 00 001252'
439 000414' 201 02 0 00 001321'
440 000415' 137 01 0 00 001271'
441 000416' 260 17 0 00 000644'
442 000417' 260 17 0 00 000631'
443
444 000420' 200 01 0 00 002234'
445 000421' 137 01 0 00 001300'
446
447
448
449 000422' 200 01 0 00 002235'
450 000423' 137 01 0 00 001305'
451 000424' 242 01 0 00 777770'
452 000425' 137 01 0 00 001306'
453
454 000426' 200 01 0 00 002232'
455
456
457
458 000427' 305 01 0 00 000000'
459 000430' 254 00 0 00 000471'
460 000431' 303 01 0 00 000003'
461 000432' 254 00 0 00 000471'
462 000433' 137 01 0 00 001275'
463
464
465
466 000434' 302 01 0 00 000000'
467 000435' 254 00 0 00 000440'
468 000436' 200 01 0 00 002233'
469 000437' 137 01 0 00 001276'

;Generate Datagram
GENDGM: $SAVE <S1,S2,T1>          ; SAVE AC'S
      $CALL CLKCTP                ; CLEAR CTPPKT AREA.
      MOVEI S2,CTPPKT
      DPB S1,CTPOP                ; OP CODE TO CTPPKT.
      $CALL NXTREF                ; GET NEXT REF NUMBER.
      $CALL REFNUM                ; STORE REF NUMBER IN CTP

      MOVE S1,DELAY
      DPB S1,DELAY1                ; STORE DELAY IN CTPPKT.

;STORE REPEAT COUNT IN CTPPKT.
      MOVE S1,RCOUNT
      DPB S1,RCNT1
      LSH S1,-8
      DPB S1,RCNT2

      MOVE S1,GENFUN                ; GENFUN TO S1.

;TEST GENFUN = 0 TO 3.
      CAIGE S1,0                    ; GENFUN >= 0.
      JRST GNDGER                  ; NO, GO TO PRINT ERROR.
      CAILE S1,3                    ; GENFUN <= 3.
      JRST GNDGER                  ; NO, GO TO PRINT ERROR.
      DPB S1,GNFNC1                ; GENFUN TO GENMSG CTPPKT.

;(IF CTP$GENFUNCT=CTP$GENFILL,THEN GENCONST TO CTPPKT.
      CAIE S1,0                    ; TEST IF GENFUNCT=GENFILL.
      JRST GNDGM1                  ; NO, CONTINUE W/O GENCONST.
      MOVE S1,GNCNST
      DPB S1,GCNST1                ; YES, LOAD CTPPKT WITH CONSTANT.
  
```



```

470
471      ;SET ZEROS IN TWO BYTES
472
473      000440' 400 01 0 00 000000      GNDGM1: SETZ      S1,
474      000441' 137 01 0 00 001277'      DPB          S1,BMAPB
475
476      000442' 200 01 0 00 002237'      MOVE          S1,GENLEN
477      000443' 137 01 0 00 001307'      DPB          S1,GNLEN1      ; STORE DELAY IN CTPPKT.
478      000444' 242 01 0 00 777770'      LSH          S1,-8          ; SHIFT M/S BYTE TO L/S BYTE
479      000445' 137 01 0 00 001310'      DPB          S1,GNLEN2      ; STORE DELAY IN CTPPKT.
480
481      ;BUILD IMAGE BUFFER (LIMITED TO 4 BYTES).
482      ;DATA LOADED INTO "IMAGDT".
483
484      000446' 200 01 0 00 002232'      MOVE          S1,GENFUN      ; TEST IF GENFUNCT=GENFIMAGE
485      000447' 302 01 0 00 000003'      CAIE         S1,3
486      000450' 254 00 0 00 000376'      JRST         GNMGEX      ; GENFUNCT NOT EQUAL GENFIMAGE,SKIP.
487
488      000451' 200 01 0 00 000351*      MOVE          S1,IMAGDT
489      000452' 242 01 0 00 777774'      LSH          S1,-4          ; RIGHT SHIFT TO RIGHT JUSTIFY.
490      000453' 137 01 0 00 001311'      DPB          S1,IMGDT1      ; STORE 1ST IMAGE BYTE IN CTPPKT.
491      000454' 242 01 0 00 777770'      LSH          S1,-8          ; SHIFT 2ND BYTE TO 1ST BYTE POSITION.
492      000455' 137 01 0 00 001312'      DPB          S1,IMGDT2      ; STORE 2ND IMAGE BYTE IN CTPPKT.
493      000456' 242 01 0 00 777770'      LSH          S1,-8          ; SHIFT 3RD BYTE TO 1ST BYTE POSITION.
494      000457' 137 01 0 00 001313'      DPB          S1,IMGDT3      ; STORE 3RD IMAGE BYTE IN CTPPKT.
495      000460' 242 01 0 00 777770'      LSH          S1,-8          ; SHIFT 4TH BYTE TO 1ST BYTE POSITION.
496      000461' 137 01 0 00 001314'      DPB          S1,IMGDT4      ; STORE 4TH IMAGE BYTE IN CTPPKT.
497      000462' 201 01 0 00 000022'      MOVEI        S1,22          ; length of ctp packet=22
498      000463' 202 01 0 00 002216'      MOVEM        S1,CTPLEN
499      000464' 254 00 0 00 000400'      JRST         GNMGE1
500
501
502      000465' 201 01 0 00 000020      GNDGEX: MOVEI   S1,20          ; SAVE LENGTH OF CTP
503      000466' 202 01 0 00 002216'      MOVEM        S1,CTPLEN      ; FOR SCSCMD PACKET.
504      000467' 260 17 0 00 001262'      GNDGE1: $CALL  REQSAV      ; SAVE CTP REQUEST
505      000470' 263 17 0 00 000000      $RET
506
507      ;ERROR PRINTOUT ROUTINES
508
509      000471' 260 17 0 00 000402*      GNDGER: $TEXT  (,<? Generate function not valid on generate datagram command>)
510      000473' 254 00 0 00 000465'      JRST         GNDGEX
  
```



```

511
512      ;Configuration
513
514      000474' 304 00 0 00 000000' CONFIG: $SAVE <S1,S2,T1>      : SAVE AC'S
515      000502' 260 17 0 00 001252' $CALL CLRCTP      : CLEAR CTPPKT AREA.
516      000503' 201 02 0 00 001321' MOVEI S2,CTPPKT
517      000504' 137 01 0 00 001271' DPB S1,CTPOP      : OP CODE TO CTPPKT.
518      000505' 260 17 0 00 000644' $CALL NXTREF      : GET NEXT REF NUMBER.
519      000506' 260 17 0 00 000631' $CALL REFNUM      : STORE REF NUMBER IN CTP
520
521      000507' 200 01 0 00 002226' MOVE S1,OTHNOD
522      000510' 137 01 0 00 001303' DPB S1,OTHND1      : STORE OTHERNODE IN CTPPKT.
523
524      000511' 201 01 0 00 000013' CNFGEX: MOVEI S1,13      : SAVE LENGTH OF CTP
525      000512' 202 01 0 00 002216' MOVEM S1,CTPLEN      : FOR SCSCMD PACKET.
526      000513' 260 17 0 00 001262' $CALL REQSAV      : SAVE CTP REQUEST
527      000514' 263 17 0 00 000000' $RET
    
```

```

528
529
530      ;Read counters
531 000515' 304 00 0 00 000000  CNTRD: $SAVE <S1,S2,T1>      : SAVE AC'S
532 000523' 260 17 0 00 001252'  $CALL CLRCTP      : CLEAR CTPPKT AREA.
533 000524' 201 02 0 00 001321'  MOVEI S2,CTPPKT
534 000525' 137 01 0 00 001271'  DPB S1,CTPOP      : OP CODE TO CTPPKT.
535 000526' 260 17 0 00 000644'  $CALL NXTREF      : GET NEXT REF NUMBER.
536 000527' 260 17 0 00 000631'  $CALL REFNUM      : STORE REF NUMBER IN CTP
537
538 000530' 200 01 0 00 002226'  MOVE S1,OTHNOD
539 000531' 137 01 0 00 001303'  DPB S1,OTHND1      : STORE OTHERNODE IN CTPPKT.
540
541 000532' 201 01 0 00 000013  CNTREX: MOVEI S1,13      : SAVE LENGTH OF CTP
542 000533' 202 01 0 00 002216'  MOVEM S1,CTPLEN      : FOR SCSCMD PACKET.
543 000534' 260 17 0 00 001262'  $CALL REQSAV      : SAVE CTP REQUEST
544 000535' 263 17 0 00 000000  $RET
  
```

```
545
546 ;Connect
547
548 000536' 304 00 0 00 000000 CONNCT: $SAVE <S1,S2,T1> ; SAVE AC'S
549 000544' 260 17 0 00 001252' $CALL CLRCTP ; CLEAR CTPPKT AREA.
550 000545' 201 02 0 00 001321' MOVEI S2,CTPPKT
551 000546' 137 01 0 00 001271' DPB S1,CTPOP ; OP CODE TO CTPPKT.
552 000547' 260 17 0 00 000644' $CALL NXTREF ; GET NEXT REF NUMBER.
553 000550' 260 17 0 00 000631' $CALL REFNUM ; STORE REF NUMBER IN CTP
554
555 000551' 200 01 0 00 002226' MOVE S1,OTHNOD
556 000552' 137 01 0 00 001303' DPB S1,OTHND1 ; STORE OTHERNODE IN CTPPKT.
557
558 000553' 201 01 0 00 000013 CONNEX: MOVEI S1,13 ; SAVE LENGTH OF CTP
559 000554' 202 01 0 00 002216' MOVEM S1,CTPLEN ; FOR SCSCMD PACKET.
560 000555' 260 17 0 00 001262' $CALL REQSAV ; SAVE CTP REQUEST
561 000556' 263 17 0 00 000000 $RET
```

```

562
563      ;Disconnect
564
565      000557' 304 00 0 00 000000      LDISC:  $SAVE  <S1,S2,T1>      ; SAVE AC'S
566      000565' 260 17 0 00 001252'      $CALL  CLRCTP      ; CLEAR CTPPKT AREA.
567      000566' 201 02 0 00 001321'      MOVEI   S2,CTPPKT      ;
568      000567' 137 01 0 00 001271'      DPB      S1,CTPOP      ; OP CODE TO CTPPKT.
569      000570' 260 17 0 00 000644'      $CALL  NXTREF      ; GET NEXT REF NUMBER.
570      000571' 260 17 0 00 000631'      $CALL  REFNUM      ; STORE REF NUMBER IN CTP
571
572      000572' 201 01 0 00 000005      DISCEX: MOVEI   S1,5      ; SAVE LENGTH OF CTP
573      000573' 202 01 0 00 002216'      MOVEM   S1,CTPLEN      ; FOR SCSCMD PACKET.
574      000574' 260 17 0 00 001262'      $CALL  REQSAV      ; SAVE CTP REQUEST
575      000575' 263 17 0 00 000000      $RET
    
```

```

576
577      ;Generate activity
578
579      000576' 304 00 0 00 000000      GENACT: $SAVE    <S1,S2,T1>      ; SAVE AC'S
580      000604' 260 17 0 00 001252'      $CALL    CLRCTP      ; CLEAR CTPPKT AREA.
581      000605' 201 02 0 00 001321'      MOVEI    S2,CTPPKT
582      000606' 137 01 0 00 001271'      DPB      S1,CTPOP      ; OP CODE TO CTPPKT.
583      000607' 260 17 0 00 000644'      $CALL    NXTREF      ; GET NEXT REF NUMBER.
584      000610' 260 17 0 00 000631'      $CALL    REFNUM      ; STORE REF NUMBER IN CTP
585
586      000611' 200 01 0 00 002234'      MOVE     S1,DELAY
587      000612' 137 01 0 00 001300'      DPB      S1,DELAY1      ; STORE DELAY IN CTPPKT.
588
589      ;STORE REPEAT COUNT IN CTPPKT
590
591      000613' 200 01 0 00 002235'      MOVE     S1,RCOUNT
592      000614' 137 01 0 00 001305'      DPB      S1,RCNT1      ; STORE REPEAT COUNT IN CTPPKT.
593      000615' 242 01 0 00 777770'      LSH      S1,-8
594      000616' 137 01 0 00 001306'      DPB      S1,RCNT2      ; STORE REPEAT COUNT IN CTPPKT.
595
596
597      000617' 200 01 0 00 002241'      MOVE     S1,ABUFNM
598      000620' 242 01 0 00 000004'      LSH      S1,4
599      000621' 202 01 0 00 001315'      MOVEM    S1,ABFNM1
600
601      000622' 200 01 0 00 002242'      MOVE     S1,ANMREQ
602      000623' 242 01 0 00 000004'      LSH      S1,4
603      000624' 202 01 0 00 001316'      MOVEM    S1,ANMRQ1
604
605      000625' 201 01 0 00 000015'      GACTEX: MOVEI    S1,15      ; SAVE LENGTH OF CTP
606      000626' 202 01 0 00 002216'      MOVEM    S1,CTPLEN      ; FOR SCSCMD PACKET.
607      000627' 260 17 0 00 001262'      $CALL    REQSAV      ; SAVE CTP REQUEST
608      000630' 263 17 0 00 000000      $RET
  
```



```

609
610 ;Store reference number given in S1.
611
612 000631' 260 17 0 00 000000* REFNUM: $SAVE <P1,P2> ;Save a couple
613 000632' 202 01 0 00 000007 MOVEM S1,P1 ; PUT REFNUM IN ACCUMULATOR.
614 000633' 402 00 0 00 000001 SETZM S1 ; INIT LOOP CNTR.
615 000634' 200 10 0 00 001272' MOVE P2,CTPREF ; SET UP POINTER.
616 000635' 241 07 0 00 000014 ROT P1,^D12 ; ALIGN FIRST (MSB) BYTE.
617
618 000636' 136 07 0 00 000010 REFNM1: IDPB P1,P2 ; DEPOSIT BYTE
619 000637' 241 07 0 00 000010 ROT P1,8 ; SHIFT DATA
620 000640' 350 00 0 00 000001 AOS S1 ; INCREMENT COUNT
621 000641' 305 01 0 00 000004 CAIGE S1,4 ; SKIP IF DONE
622 000642' 254 00 0 00 000636' JRST REFNM1 ; NOT DONE, LOOP
623 000643' 263 17 0 00 000000 $RET
624
625 ;Get next reference number: From Ryan's code.
626
627 000644' 350 01 0 00 002246' NXTREF: AOS S1,NUMREF ; INCREMENT REF NUMBER BASE
628 000645' 607 01 0 00 040000 TLNN S1,1B21 ; HAS IT REACHED IT'S MAX VALUE
629 000646' 263 17 0 00 000000 $RET ; NO, RETURN TO CALLER.
630 000647' 402 00 0 00 002246' SETZM NUMREF ; YES, INIT IT.
631 000650' 254 00 0 00 000644' JRST NXTREF ; GO BACK AN START COUNTING AGIAN.
  
```

```

632          SUBTTL  Send Message/Datagram
633
634          ;BUILD SCS PACKET AND LOAD S1=FUNCTION (SSSMG=SEND MESSAGE)
635          ;AND S2=ADDRESS OF SCS PACKET.
636
637          000651' 304 00 0 00 000000      SNDMSG: $SAVE    <S1>          ;Save an AC
638          000655' 201 01 0 00 000006      MOVEI    S1,.SSSMG      ;Get send message argument
639          000656' 260 17 0 00 000667'      $CALL    SND1          ;Send the message
640          000657' 263 17 0 00 000000      $RET          ;Return to code that called this code
641
642          ;BUILD SCS PACKET AND LOAD S1=FUNCTION (SSSDG=SEND DATAGRAM)
643          ;AND S2=ADDRESS OF SCS PACKET.
644
645          000660' 304 00 0 00 000000      SNDDAT: $SAVE    <S1>          ;Save an AC
646          000664' 201 01 0 00 000004      MOVEI    S1,.SSSDG      ;Get send datagram argument
647          000665' 260 17 0 00 000667'      $CALL    SND1          ;Send the datagram
648          000666' 263 17 0 00 000000      $RET          ;Return to code that called this code
649
650          ;Call this subroutine with S1 having either .SSSDG, the send datagram argument
651          ; or .SSSMG, the send message argument.
652
653          000667' 304 00 0 00 000000      SND1:  $SAVE    <S1,S2,T1>      ;Save acs
654
655          000675' 260 17 0 00 001242'      $CALL    CLRCMD          ;Clear SCSCMD buffer
656          000676' 402 00 0 00 002215'      SETZM    SUCESS
657
658          000677' 201 02 0 00 001401'      MOVEI    S2,SCSCMD      ;Point to the buffer
659
660          000700' 201 03 0 00 000005      MOVEI    T1,.LBSMG          ;Load length of SSSMG/SSSDG
661          000701' 202 03 0 02 000000      MCVEM    T1,.SQLEN(S2)
662
663          000702' 200 03 0 00 000000*      MOVE     T1,CONID          ;Load connect ID
664          000703' 202 03 0 02 000001      MOVEM    T1,.SQCID(S2)
665
666          000704' 201 03 0 00 001321'      MOVEI    T1,CTPPKT          ;Point to CTP packet
667          000705' 202 03 0 02 000002      MOVEM    T1,.SQAPT(S2)
668
669          000706' 200 03 0 00 002216'      MOVE     T1,CTPLEN          ;Load length of CTP packet
670          000707' 202 03 0 02 000003      MOVEM    T1,.SQLPT(S2)
671
672          000710' 200 03 0 00 002220'      MOVE     T1,FLAGS          ;Get flags
673          000711' 434 03 0 00 000000*      IOR      T1,PATHS          ;Add in path selection
674          000712' 202 03 0 02 000004      MOVEM    T1,.SQFLG(S2)      ;Store
675
676          000713' 260 17 0 00 000000*      $CALL    SCSS$          ;Perform SCS JSYS or UUD
677          000714' 260 17 0 00 000000*      $CALL    SCSERR          ;Output error message
678          000715' 350 00 0 00 002215'      AOS      SUCESS
679          000716' 263 17 0 00 000000      $RET
  
```

```

680
681      ;RETURN THE FIRST ENTRY FROM THE DATA REQUEST COMPLETE QUEUE.
682
683      000717' 304 00 0 00 000000      DMARD:  $SAVE    <S1,S2,T1,T2>      ; SAVE AC'S
684      000726' 201 01 0 00 000024      MOVEI    S1,..SSGDE      ;Get read message argument
685
686      000727' 402 00 0 00 002215'      SETZM    SUCESS      ; RESET SUCCESS STATUS.
687
688      000730' 201 02 0 00 001461'      MOVEI    S2,SCSRSP
689
690      000731' 200 03 0 00 000702*      MOVE     T1,CONID
691      000732' 202 03 0 02 000001      MOVEM    T1,1(S2)
692
693      000733' 260 17 0 00 000713*      $CALL    SCS$      ;Perform SCS JSYS or UUO
694      000734' 260 17 0 00 000714*      $CALL    SCSERR      ;Output error message
695      000735' 350 00 0 00 002215'      AOS      SUCESS
696
697      000736' 200 01 0 02 000002      MOVE     S1,..SQARB(S2)      ; GET ADRS RETURNED BUFFER
698      000737' 202 01 0 00 000741'      MOVEM    S1,DMADRS      ; SAVE IN RETBUF.
699      000740' 263 17 0 00 000000      $RET      ;Return to code that called this code
700
701      000741' 000000 000000      DMADRS: 0
  
```

```

702          SUBTTL  Read Message/Datagram
703
704          ;Read message and Read datagram subroutines.
705
706 000742' 304 00 0 00 000000 READM: $SAVE    <S1>          ;Save an AC
707 000746' 201 01 0 00 000013      MOVEI    S1,.SSRMG      ;Get read message argument
708 000747' 260 17 0 00 000760'      $CALL    READMA      ;Read the message
709 000750' 263 17 0 00 000000      $RET          ;Return to code that called this code
710
711 000751' 304 00 0 00 000000 READD: $SAVE    <S1>          ;Save an AC
712 000755' 201 01 0 00 000022      MOVEI    S1,.SSRDG      ;Get read datagram argument
713 000756' 260 17 0 00 000760'      $CALL    READMA      ;Read the datagram
714 000757' 263 17 0 00 000000      $RET          ;Return to code that called this code
715
716          ;Call with S1 having either .SSRMG, the read message argument or .SSRDG, the
717          ; read datagram argument.
718
719 000760' 304 00 0 00 000000 READMA: $SAVE    <S1,S2,T1,T2,T3>      ; SAVE AC'S
720
721 000770' 402 00 0 00 002215'      SETZM     SUCESS          ; RESET SUCCESS STATUS.
722
723 000771' 201 02 0 00 001461'      MOVEI     S2,SCSRSP
724
725 000772' 201 03 0 00 000005      MOVEI     T1,.LBRMG
726 000773' 202 03 0 02 000000      MOVEM     T1,(S2)
727
728 000774' 200 03 0 00 000731*      MOVE      T1,CONID
729 000775' 202 03 0 02 000001      MOVEM     T1,1(S2)
730
731 000776' 260 17 0 00 000733*      $CALL     SCSS$          ;Perform SCS JSYS or UUO
732 000777' 260 17 0 00 000734*      $CALL     SCSERR        ;Output error message
733 001000' 350 00 0 00 002215'      AOS        SUCESS
734
735 001001' 200 01 0 02 000002      MOVE      S1,.SQBID(S2)      ; GET ADRS RETURNED BUFFER
736 001002' 202 01 0 00 002243'      MOVEM     S1,RETBUF        ; SAVE IN RETBUF.
737
738 001003' 200 02 0 02 000004      MOVE      S2,.SQLRP(S2)      ; GET LENGTH OF RETURNED BUFFER
739 001004' 202 02 0 00 002244'      MOVEM     S2,RLENBF        ; SAVE IN RLENBF
740
741 001005' 515 03 0 01 000000      HRLZI     T1,0(S1)
742 001006' 541 03 0 00 001621'      HRRI      T1,SAVRSP
743 001007' 251 03 0 00 001667'      BLT       T1,SAVRSP+46
744
745          ;TEST FOR OPCODE OF RESPONSE PACKET.
746
747 001010' 402 00 0 00 000005      SETZM     T3              ; CLEAR T3
748 001011' 200 02 0 00 002243'      MOVE      S2,RETBUF        ; ADRS OF RSP BUFFER
749 001012' 135 05 0 00 001317'      LDB       T3,OPSAV1        ; T3=RSP OP CODE.
  
```



```

750                                     ;TEST FOR RESPONSE OPCODE = 64 (FUNCTION SET RESPONSE).
751
752
753 001013' 302 05 0 00 000100      CAIE    T3,100          : TEST IF BMAP RSPNS
754 001014' 254 00 0 00 001065'      JKST    READM1         : JUMP IF NOT BMAP RSP
755 001015' 120 03 0 02 000001      DMOVE   T1,1(S2)       : MOVE 1ST 6 BYTES->T1,T2
756 001016' 242 03 0 00 777774      LSH     T1,-4          : RIGHT JUSTIFY 1ST 2 BYTES.
757
758 001017' 246 03 0 00 000020      LSHC    T1,^D16        : RIGHT JUSTIFY 4 BYTES.
759 001020' 202 03 0 00 002201'      MOVEM   T1,RSP0A       : MOVE 1ST 4 BYTES TO RSP0A.
760 001021' 402 00 0 00 000003      SETZM   T1              : CLEAR T1.
761 001022' 246 03 0 00 000016      LSHC    T1,16         : BYTES 4,5 TO T1.
762 001023' 200 04 0 02 000003      MOVE     T2,3(S2)       : BYTES 6-9 TO T2.
763
764 001024' 246 03 0 00 000020      LSHC    T1,^D16        : RIGHT JUSTIFY 4 BYTES.
765 001025' 202 03 0 00 002202'      MOVEM   T1,RSP0A+1     : MOVE 2ND 4 BYTES TO RSP0A+1.
766 001026' 402 00 0 00 000003      SETZM   T1              : CLEAR T1.
767 001027' 246 03 0 00 000016      LSHC    T1,16         : BYTES 8,9 TO T1.
768 001030' 200 04 0 02 000004      MOVE     T2,4(S2)       : BYTES 10-13 TO T2.
769
770 001031' 246 03 0 00 000020      LSHC    T1,^D16        : RIGHT JUSTIFY 4 BYTES.
771 001032' 202 03 0 00 002203'      MOVEM   T1,RSP0A+2     : MOVE 3RD 4 BYTES TO RSP0A+2.
772 001033' 402 00 0 00 000003      SETZM   T1              : CLEAR T1.
773 001034' 246 03 0 00 000016      LSHC    T1,16         : BYTES 12,13 TO T1.
774 001035' 200 04 0 02 000005      MOVE     T2,5(S2)       : BYTES 14-17 TO T2.
775
776 001036' 246 03 0 00 000020      LSHC    T1,^D16        : RIGHT JUSTIFY 4 BYTES.
777 001037' 202 03 0 00 002204'      MOVEM   T1,RSP0A+3     : MOVE 4TH 4 BYTES TO RSP0A+3.
778 001040' 402 00 0 00 000003      SETZM   T1              : CLEAR T1.
779 001041' 246 03 0 00 000016      LSHC    T1,16         : BYTES 16,17 TO T1.
780 001042' 200 04 0 02 000006      MOVE     T2,6(S2)       : BYTES 18-21 TO T2.
781
782 001043' 246 03 0 00 000020      LSHC    T1,^D16        : RIGHT JUSTIFY 4 BYTES.
783 001044' 202 03 0 00 002205'      MOVEM   T1,RSP0A+4     : MOVE 5TH 4 BYTES TO RSP0A+4.
784 001045' 402 00 0 00 000003      SETZM   T1              : CLEAR T1.
785 001046' 246 03 0 00 000016      LSHC    T1,16         : BYTES 20,21 TO T1.
786 001047' 200 04 0 02 000007      MOVE     T2,7(S2)       : BYTES 22-25 TO T2.
787
788 001050' 246 03 0 00 000020      LSHC    T1,^D16        : RIGHT JUSTIFY 4 BYTES.
789 001051' 202 03 0 00 002206'      MOVEM   T1,RSP0A+5     : MOVE 6TH 4 BYTES TO RSP0A+5.
790 001052' 402 00 0 00 000003      SETZM   T1              : CLEAR T1.
791 001053' 246 03 0 00 000016      LSHC    T1,16         : BYTES 24,25 TO T1.
792 001054' 200 04 0 02 000010      MOVE     T2,8(S2)       : BYTES 26-29 TO T2.
793
794 001055' 246 03 0 00 000020      LSHC    T1,^D16        : RIGHT JUSTIFY 4 BYTES.
795 001056' 202 03 0 00 002207'      MOVEM   T1,RSP0A+6     : MOVE 7TH 4 BYTES TO RSP0A+6.
796 001057' 402 00 0 00 000003      SETZM   T1              : CLEAR T1.
797 001060' 246 03 0 00 000016      LSHC    T1,16         : BYTES 28,29 TO T1.
798 001061' 200 04 0 02 000011      MOVE     T2,9(S2)       : BYTES 30-31 TO T2.
799
800 001062' 246 03 0 00 000020      LSHC    T1,^D16        : RIGHT JUSTIFY 4 BYTES.
801 001063' 202 03 0 00 002210'      MOVEM   T1,RSP0A+7     : MOVE 8TH 4 BYTES TO RSP0A+7.
802 001064' 254 00 0 00 001075'      JRST    READMX
  
```



```

803
804 001065' 302 05 0 00 000101 READM1: CAIE T3,101 ; TEST IF BMAP RSPNS
805 001066' 254 00 0 00 001075' JRST READMX ; JUMP IF NOT BMAP RSP
806 001067' 200 01 0 02 000004' MOVE S1,4(S2) ; LOCAL BUFFER # TO S1
807 001070' 202 01 0 00 002213' MOVEM S1,BUFLNM ; SAVE IN BUFLNM
808 001071' 242 01 0 0C 777774' LSH S1,-4 ; RIGHT JUSTIFY VAX BFRNAM
809 001072' 260 17 0 00 001100' $CALL P11T10 ; SWAP VAX BFRNAM TO 10 BUFRNAM
810 001073' 201 02 0 00 001077' MOVEI S2,BFLNM1 ; S2=10 BUFFER NAME LOC.
811 001074' 137 01 0 00 001076' DPB S1,BFR10 ; SAVE IN 10 BUFFER NAME LOC.
812
813 001075' 263 17 0 00 00C000 READMX: $RET
814
815 001076' 04 40 0 02 000000 BFR10: POINT 32,(S2),31 ; 4 BYTE FOR 10 STYLE BUFRNAM.
816 001077' 000000 000000 BFLNM1: C
817
818 ;THIS ROUTINE CHANGES A 4X8BIT BYTE PDP11 RIGHT JUSTIFIED WORD
819 ;TO A 4X8 BIT BYTE PDP10 WORD.
820 ;!BYTE0!BYTE1!BYTE2!BYTE3! => !BYTE3!BYTE2!BYTE1!BYTE0!
821 ;THIS ROUTINE IS CALLED BY:
822 ; MOVEI S1,[N] ;[N]=PDP11 WORD.
823 ; $CALL P11T10 ;
824 ; +1 ;S1 HAS CHANGED WORD.
825
826 001100' 304 00 0 00 000000 P11T10: $SAVE <S2,T1> ;Save a couple of ACs
827 001105' 402 00 0 00 000002 SETZM S2 ;Clear result
828 001106' 135 03 0 00 002516' LDB T1,[POINT 8,S1,35]
829 001107' 137 03 0 00 002517' DPB T1,[POINT 8,S2,11]
830 001110' 135 03 0 00 002520' LDB T1,[POINT 8,S1,27]
831 001111' 137 03 0 00 002521' DPB T1,[POINT 8,S2,19]
832 001112' 135 03 0 00 002522' LDB T1,[POINT 8,S1,19]
833 001113' 137 03 0 00 002523' DPB T1,[POINT 8,S2,27]
834 001114' 135 03 0 00 002524' LDB T1,[POINT 8,S1,11]
835 001115' 137 03 0 00 002525' DPB T1,[POINT 8,S2,35]
836 001116' 202 02 0 00 000001 MOVEM S2,S1 ;Return result in S1
837 001117' 263 17 0 00 000000 $RET
    
```

```

838          SUBTTL Requeue Message/Datagram Buffer
839
840          ;Requeue message and datagram buffer.
841
842          ;Requeue message buffer
843
844          001120' 304 00 0 00 000000    REQUEM: $SAVE    <S1>          ;Save an AC
845          001124' 201 01 0 00 000007    MOVEI    S1,SSQRM      ;Get queue message buffer argument
846          001125' 260 17 0 00 001136'    $CALL    RQIUEU      ;Queue the message buffer
847          001126' 263 17 0 00 000000    $RET          ;Return to code that called this code
848
849          ;Requeue datagram buffer
850
851          001127' 304 00 0 00 000000    REQUED: $SAVE    <S1>          ;Save an AC
852          001133' 201 01 0 00 000005    MOVEI    S1,SSQRD      ;Get queue datagram buffer argument
853          001134' 260 17 0 00 001136'    $CALL    RQUEUE      ;Queue the datagram buffer
854          001135' 263 17 0 00 000000    $RET          ;Return to code that called this code
855
856          ;Routine to do the work, can't touch S1 for awhile.
857
858          001136' 304 00 0 00 000000    RQUEUE: $SAVE    <S1,S2,T1>      ; SAVE AC'S
859
860          001144' 201 02 0 00 001401'    MOVEI    S2,SCSCMD      ;Point to block
861
862          001145' 201 03 0 00 000003    MOVEI    T1,LBQRM      ;Set length
863          001146' 202 03 0 02 000000    MOVEM    T1,(S2)
864
865          001147' 200 03 0 00 000774*    MOVE     T1,CONID      ;Set connect ID
866          001150' 202 03 0 02 000001    MOVEM    T1,1(S2)
867
868          001151' 201 03 0 00 002243'    MOVE     T1,RETBUF      ; SAVE ADRS OF BUFFER CHAIN
869          001152' 202 03 0 02 000002    MOVEM    T1,2(S2)      ; TO QUEUE
870          001153' 402 00 1 00 002243'    SETZM    @RETBUF
871
872          001154' 260 17 0 00 000776*    $CALL    SCSS$        ;Perform SCS JSYS or UUO
873          001155' 260 17 0 00 000777*    $CALL    SCSERR       ;Output error message
874          001156' 350 00 0 00 002215'    AOS      SUCESS
875          001157' 263 17 0 00 000000    $RET
  
```

```

876                      SUBTTL Subroutines
877
878                      ;Examine message
879
880 001160' 304 00 0 00 000000    EXAMNM: $SAVE    <S1,S2,T1>          ;Save ACs
881
882 001166' 201 02 0 00 001701'    MOVE!    S2,SAVREQ
883 001167' 135 01 0 00 001317'    LDB      S1,OPSAV1
884 001170' 200 02 0 00 002243'    MOVE     S2,RETBUF
885 001171' 135 03 0 00 001317'    LDB      T1,OPSAV1
886                                $TEXT    (,<
887                                Request op code = ^0/S1/
888                                Response op code = ^0/T1/>)
889 001174' 200 01 0 00 001701'    MOVE     S1,SAVREQ
890 001175' 200 02 0 00 001702'    MOVE     S2,SAVREQ+1
891 001176' 242 01 0 00 777774      LSH      S1,-4          ; RIGHT JUSTIFY 3 BYTES OF REF.
892 001177' 246 01 0 00 000010      LSHC     S1,8          ; L/S LSB INTO S1 FROM S2
893 001200' 200 02 0 00 002250'    MOVE     S2,REFMSK      ; S2=MASK FOR REF NUMBER.
894 001201' 404 01 0 00 000002      AND       S1,S2          ; S1 WILL HAVE MASKED REF NUMBER.
895 001202' 260 17 0 00 001172*    $TEXT    (,<Request reference number = ^0/S1/>)
896 001204' 200 03 0 00 002243'    MOVE     T1,RETBUF      ; T1=ADRS OF RESPONSE PACKET.
897 001205' 200 01 0 03 C00000      MOVE     S1,(T1)        ; S1=1ST SECTION OF REFNUM.
898 001206' 200 02 0 03 000001      MOVE     S2,1(T1)       ; S2=2ND SECTION OF REFNUM.
899 001207' 242 01 0 00 777774      LSH      S1,-4          ; RIGHT JUSTIFY 3 BYTES OF REF.
900 001210' 246 01 0 00 000010      LSHC     S1,8          ; L/S LSB INTO S1 FROM S2
901 001211' 200 02 0 00 002250'    MOVE     S2,REFMSK      ; S2=MASK FOR REF NUMBER.
902 001212' 404 01 0 00 000002      AND       S1,S2          ; S1 WILL HAVE MASKED REF NUMBER.
903 001213' 260 17 0 00 001202*    $TEXT    (,<Response reference number = ^0/S1/>)
904 001215' 200 02 0 00 002243'    MOVE     S2,RETBUF      ;Point to buffer
905 001216' 135 01 0 00 001320'    LDB      S1,STATSV     ;Get status
906 001217' 260 17 0 00 001213*    $TEXT    (,<Response status = ^0/S1/>)
907 001221' 263 17 0 00 000000      $RET

```

908
 909
 910
 911
 912
 913
 914
 915
 916
 917
 918
 919
 920
 921
 922
 923
 924
 925
 926
 927
 928
 929

001222' 304 00 0 00 000000
 001227' 402 00 0 00 000002
 001230' 135 03 0 00 002516'
 001231' 137 03 0 00 002613'
 001232' 135 03 0 00 002520'
 001233' 137 03 0 00 002614'
 001234' 135 03 0 00 002522'
 001235' 137 03 0 00 002615'
 001236' 135 03 0 00 002524'
 001237' 137 03 0 00 002616'
 001240' 200 01 0 00 000002
 001241' 263 17 0 00 000000

;RVSNUM-- THIS ROUTINE REVERSES THE ORDER OF BYTES(8 BIT) WITHIN A WORD.
 ;IT ACCEPTS A RIGHT JUSTIFIED 36 BIT NUMBER IN S1 AND RETURNS IT TO THE
 ;USER WITH THE LS BYTE MOVED TO THE MS BYTE,ETC.

;CALLING SEQ.
 : MOVE S1,ARG
 : SCALL RVSNUM

RVSNUM: \$SAVE <S2,T1>
 SETZM S2
 LDB T1,[POINT 8,S1,35]
 DPB T1,[POINT 8,S2,7]
 LDB T1,[POINT 8,S1,27]
 DPB T1,[POINT 8,S2,15]
 LDB T1,[POINT 8,S1,19]
 DPB T1,[POINT 8,S2,23]
 LDB T1,[POINT 8,S1,11]
 DPB T1,[POINT 8,S2,31]
 MOVE S1,S2
 \$RET

;Save some ACs
 ;CLR FOR TEMP STORAGE.
 ;MOVE LS BYTE OF S1 TO S2 START AT MS
 ;MOVE LS BYTE OF S1 TO S2 START AT MS
 ;MOVE LS BYTE OF S1 TO S2 START AT MS
 ;MOVE LS BYTE OF S1 TO S2 START AT MS
 ;GET THE REVERSED WORD FOR THE CALLER

```

930
931      ;Clear SCSCMD buffer for building arguments
932
933      001242' 304 00 0 00 000000      CLRCMD: $SAVE    <S1>          ;Save an AC
934      001246' 402 00 0 00 001401'      SETZM      SCSCMD          ; CLR 1ST LOCATION OF ARG BLK
935      001247' 200 01 0 00 002617'      MOVE       S1,[SCSCMD,,SCSCMD+1] ; SET POINTERS IN S1
936      001250' 251 01 0 00 001447'      BLT        S1,SCSCMD+46      ; CLR REST OF ARG BLK
937      001251' 263 17 0 00 000000      $RET
938
939      ;Clear CTPPKT area.
940
941      001252' 304 00 0 00 000000      CLRCTP: $SAVE    <S1>          ;Save an AC
942      001256' 402 00 0 00 001321'      SETZM      CTPPKT          ; CLR 1ST LOCATION OF ARG BLK
943      001257' 200 01 0 00 002620'      MOVE       S1,[CTPPKT,,CTPPKT+1] ; SET POINTERS IN S1
944      001260' 251 01 0 00 001367'      BLT        S1,CTPPKT+46      ; CLR REST OF ARG BLK
945      001261' 263 17 0 00 000000      $RET
946
947      ;Copy CTPPKT to SAVREQ for later debugging examination.
948
949      001262' 304 00 0 00 000000      REQSAV: $SAVE    <S1>          ;Save an AC
950      001266' 200 01 0 00 002621'      MOVE       S1,[CTPPKT,,SAVREQ] ;Load LBT pointer
951      001267' 251 01 0 00 001747'      BLT        S1,SAVREQ+46      ;Move the data
952      001270' 263 17 0 00 000000      $RET

```



```

953          SUBTTL Field Byte Pointers
954
955          ;COMMON TO ALL COMMANDS
956
957 001271' 34 10 0 02 000000  CTPOP: POINT 8,(S2),7
958 001272' 34 10 0 02 000000  CTPREF: POINT 8,(S2),7
959
960          ;UNIQUE TO BUFFER MAP COMMAND.
961
962 001273' 24 10 0 02 000001  BFTYP1: POINT 8,1(S2),15
963 001274' 04 20 0 02 000001  BMAPA: POINT 16,1(S2),31
964 001275' 34 10 0 02 000002  GNFNC1: POINT 8,2(S2),7
965 001276' 24 10 0 02 000002  GCNST1: POINT 8,2(S2),15
966 001277' 04 20 0 02 000002  BMAPB: POINT 16,2(S2),31
967 001300' 24 10 0 02 000001  DELAY1: POINT 8,1(S2),15
968 001301' 34 10 0 02 000002  MVTYP1: POINT 8,2(S2),07
969 001302' 14 10 0 02 000002  PKTSZ1: POINT 8,2(S2),23
970 001303' 24 10 0 02 000002  OTHND1: POINT 8,2(S2),15
971 001304' 04 10 0 02 000002  PKMLT1: POINT 8,2(S2),31
972 001305' 14 10 0 02 000001  RCNT1: POINT 8,1(S2),23
973 001306' 04 10 0 02 000001  RCNT2: POINT 8,1(S2),31
974 001307' 34 10 0 02 000003  GNLEN1: POINT 8,3(S2),7
975 001310' 24 10 0 02 000003  GNLEN2: POINT 8,3(S2),15
976 001311' 14 10 0 02 000003  IMGDT1: POINT 8,3(S2),23
977 001312' 04 10 0 02 000003  IMGDT2: POINT 8,3(S2),31
978 001313' 34 10 0 02 000004  IMGDT3: POINT 8,4(S2),07
979 001314' 24 10 0 02 000004  IMGDT4: POINT 8,4(S2),15
980 001315' 04 40 0 02 000002  ABFNM1: POINT 32,2(S2),31
981 001316' 04 40 0 02 000003  ANMRQ1: POINT 32,3(S2),31
982 001317' 34 10 0 02 000000  OPSAV1: POINT 8,(S2),7
983 001320' 24 10 0 02 000001  STATSV: POINT 8,1(S2),15
  
```

		SUBTTL	Storage
984			
985			
986	001321'	CTPPKT: BLOCK	^D38
987	001367'	BLOCK	^D10
988	001401'	SCSCMD: BLOCK	^D38
989	001447'	BLOCK	^D10
990	001461'	SCSRSP: BLOCK	^D38
991	001527'	BLOCK	^D10
992	001541'	SCSEVT: BLOCK	^D38
993	001607'	BLOCK	^D10
994			
995		;SAVE AREAS FOR REQUEST,RESPONSE PACKETS.	
996			
997	001621'	SAVRSP: BLOCK	^D38
998	001667'	BLOCK	^D10
999	001701'	SAVREQ: BLOCK	^D38
1000	001747'	BLOCK	^D10
1001			
1002	001761'	SAVCTP: BLOCK	^D38
1003	002027'	BLOCK	^D10
1004	002041'	SAVSCS: BLOCK	^D38
1005	002107'	BLOCK	^D10
1006	002121'	SCSSAV: BLOCK	^D38
1007	002167'	BLOCK	^D10
1008			
1009		;8 BYTE OR 256 BIT BLOCK FROM FUNCTION SET RESPONSE WHICH	
1010		;CORRESPONDS TO THE OPCODES WHICH ARE IMPLEMENTED	
1011		;THIS BLOCK CONTAINS DATA WHICH IS RIGHT JUSTIFIED.	
1012			
1013	002201'	RSP0A: BLOCK	^D8

```
1014
1015      ;Storage used to build packets
1016
1017 002211' 000000 000000      OPCODE: 0
1018 002212' 000000 000000      BUFLN: 0
1019 002213' 000000 000000      BUFLNM: 0
1020 002214' 000000 000000      LOFSET: 0
1021 002215' 000000 000000      SUCESS: 0
1022 002216' 000000 000000      CTPLEN: 0
1023 002217' 000000 000000      TIMER: 0
1024 002220' 000000 000000      FLAGS: 0
1025 002221' 000000 000000      MOVTYP: 0
1026 002222' 000000 000000      ACTFLG: 0
1027 002223' 000000 000000      EXTEND: 0
1028 002224' 000000 000000      STADR: 0
1029 002225' 000000 000000      PKTSIZ: 0
1030 002226' 000000 000000      OTHNOD: 0
1031 002227' 000000 000000      PKTMLT: 0
1032 002230' 000000 000000      ROFSET: 0
1033 002231' 000000 000000      BUFRNM: 0
1034 002232' 000000 000000      GENFUN: 0
1035 002233' 000000 000000      GNCNST: 0
1036 002234' 000000 000000      DELAY: 0
1037 002235' 000000 000000      PCOUNT: 0
1038 002236' 000000 000000      PKTMUL: 0
1039 002237' 000000 000000      GENLEN: 0
1040 002240' 000000 000000      IMGDAT: 0
1041 002241' 000000 000000      ABUFNM: 0
1042 002242' 000000 000000      ANMREQ: 0
1043 002243' 000000 000000      RETBUF: 0
1044 002244' 000000 000000      RLENBF: 0
1045 002245' 000000 000000      BFRNXT: 0
1046 002246' 000100 401403      NUMREF: 100,,401403
1047 002247' 000000 000000      BUFTYP: 0
1048 002250' 037777 777777      REFMSK: 37777,,777777
```

```
1049          SUBTTL Literals
1050
1051 002251'    LIT..2: XLIST          ;LIT
1052             LIST
1053
1054          END
```

NO ERRORS DETECTED

PROGRAM BREAK IS 002651
CPU TIME USED 00:28.808

150P CORE USED

SEQ 0219

[illegible]

DISCEX	572#																			
DMADRS	36	698	701#																	
DMARD	36	683#																		
EXAMNM	30	880#																		
EXTEND	35	1027#																		
FLAGS	672	1024#																		
FSET	73	100#																		
GACTEX	605#																			
GCNST1	150	357	469	965#																
GENACT	93	579#																		
GENDGM	83	437#																		
GENFUN	31	135	342	374	454	484	1034#													
GENLEN	30	33	366	476	1039#															
GENMSG	81	325#																		
GNCNST	31	149	356	468	1035#															
GNDGE1	504#																			
GNDGER	459	461	509#																	
GNDGEX	502#	510																		
GNDGM1	467	473#																		
GNFERR	140	142	179#																	
GNFNC1	143	350	462	964#																
GNLEN1	367	477	974#																	
GNLEN2	369	479	975#																	
GNMGE1	423	427#	499																	
GNMGER	347	349	432#																	
GNMGEX	376	425#	433	486																
GNMSG1	355	361#																		
GNMSG2	399#	408	411																	
GNMSG3	404	407#																		
GNMSG5	405	416#																		
HSCDST	48#																			
IMAGDT	48#	392	488																	
IMGDAT	33	1040#																		
IMGDT1	490	976#																		
IMGDT2	492	977#																		
IMGDT3	494	978#																		
IMGDT4	496	979#																		
IMGLN	48#	407																		
ITERCT	48#	403	421																	
LDISC	91	565#																		

SEQ 0221

[illegible]

RSPOA
RVSNUM
S1

SEQ 0222

S2

SAVCTP
SAVREQ
SAVRSP
SAVSCS
SCSS
SCSCMD
SCSEVT
SCSRSP
SCSSAV
SND1
SNDDAT
SNDMSG
STADR
STATSV
SUCESS
TXTEXT
T1

759	765	771	777	783	789	795	801	1013#					
169	211	279	290	295	304	918#							
70	71	72	74	76	78	80	82	84	86	88	90	92	100
103	107	108	113	116	119	120	124	126	128	132	135	135	139
141	143	147	149	150	158	159	161	163	165	166	168	170	172
173	187	190	193	197	199	201	206	207	210	212	216	217	219
220	232	235	238	239	244	245	246	247	248	252	254	256	258
262	264	266	268	269	273	274	278	280	284	285	289	291	293
294	296	299	300	301	303	305	307	308	325	328	331	333	337
338	339	340	342	346	348	350	354	356	357	361	361	366	367
368	369	374	375	389	390	420	421	422	425	426	426	440	444
445	449	450	451	452	454	458	460	462	466	468	468	473	474
476	477	478	479	484	485	488	489	490	491	492	493	494	495
496	497	498	502	503	514	517	521	522	524	525	531	534	538
539	541	542	548	551	555	556	558	559	565	568	572	573	579
582	586	587	591	592	593	594	597	598	599	601	602	603	605
606	613	614	620	621	627	628	637	638	645	646	653	683	684
697	698	706	707	711	712	719	735	736	741	806	807	808	811
828	830	832	834	836	844	845	851	852	858	880	883	889	891
892	894	897	899	900	902	905	920	922	924	926	928	933	935
936	941	943	944	949	950	951							
70	100	102	113	115	170	187	189	208	212	217	232	234	280
285	291	296	301	305	325	327	390	400	437	439	514	516	531
533	548	550	565	567	579	581	653	658	661	664	667	670	674
683	688	691	697	719	723	726	729	735	738	739	748	755	762
768	774	780	786	792	798	806	810	815	826	827	829	831	833
835	836	858	860	863	866	869	880	882	884	890	893	894	898
901	902	904	918	919	921	923	925	927	928	957	958	962	963
964	965	966	967	968	969	970	971	972	973	974	975	976	977
978	979	980	981	982	983								
34	1002#												
34	882	889	890	950	951	999#							
35	742	743	997#										
34	1004#												
40#	676	693	731	872									
34	658	860	934	935	936	988#							
40#	677	694	732	873									
34	992#												
34	688	723	990#										
34	1006#												
639	647	653#											
35	645#												
30	637#												
36	1028#												
905	983#												
656	678	686	695	721	733	874	1021#						
95	179	182	227	314	317	320	432	509	888	895	903	906	565
70	100	113	187	232	325	391	402	403	437	514	531	548	683
579	653	660	661	663	664	666	667	669	670	672	673	674	759
690	691	719	725	726	728	729	741	742	743	755	756	758	779
760	761	764	765	766	767	770	771	772	773	776	777	778	799
782	783	784	785	788	789	790	791	794	795	796	797	800	801
826	828	829	830	831	832	833	834	835	858	862	863	865	866

	868	869	880	885	896	897	898	918	920	921	922	923	924	925	SEQ 0223
T10DST	926	927													
T2	48#														
T20DST	399	400	683	719	762	768	774	780	786	792	798				
T3	48#														
T4	392	393	394	410	719	747	749	753	804						
	70	100	113	187	232	325	394	399	410	437	514	531	548	565	
	579	612	637	645	653	683	706	711	719	826	844	851	858	880	
	918	933	941	949											
TIMER	1023#														
UNMPEX	219#	228													
VEDIT	23														
VMSDST	48#														
\$RETF	27														
\$RETIF	27														
\$RETIT	27														
\$RETT	27														
..0001	71	71#													
..0002	101	101#													
..0003	114	114#													
..0004	188	188#													
..0005	233	233#													
..0006	326	326#													
..0007	438	438#													
..0010	515	515#													
..0011	532	532#													
..0012	549	549#													
..0013	566	566#													
..0014	580	580#													
..0015	613#														
..0016	638	638#													
..0017	646	646#													
..0020	654	654#													
..0021	684	684#													
..0022	707	707#													
..0023	712	712#													
..0024	720	720#													
..0025	827	827#													
..0026	845	845#													
..0027	852	852#													
..0030	859	859#													
..0031	881	881#													
..0032	919	919#													
..0033	934	934#													
..0034	942	942#													
..0035	950	950#													
..TSA1	70#	70	100#	100	113#	113	187#	187	232#	232	325#	325	437#	437	
	514#	514	531#	531	548#	548	565#	565	579#	579	612#	612	637#	637	
	645#	645	653#	653	683#	683	706#	706	711#	711	719#	719	826#	826	
	844#	844	851#	851	858#	858	880#	880	918#	918	933#	933	941#	941	
	949#	949													
.A13	70	100	113	187	232	325	437	514	531	548	565	579	613	637	
	645	653	683	706	711	719	826	844	851	858	880	918	933	941	

Label	949	100	113	187	232	325	437	514	531	548	565	579	613	637	SEQ
.A14	70 645 949	653	683	706	711	719	826	844	851	858	880	918	933	941	0224
.A15	70 645 949	653	683	706	711	719	826	844	851	858	880	918	933	941	
.A16	70 637 941	645 949	653	683	706	711	719	826	844	851	858	880	918	933	
.ACB	70# 188# 514# 566# 646# 711# 845# 918# 950#	70 188 514 566 646 711 845 918 950	71# 232# 515# 579# 653# 712# 851# 919#	71 232 515 579 653 712 851 919	100# 233# 531# 580# 654# 719# 852# 933#	100 233 531 580 654 719 852 933	101# 325# 552# 612# 683# 720# 858# 934#	101 325 532 612 683 720 858 934	113# 326# 548# 637# 684# 826# 859# 941#	113 326 548 637 684 826 859 941	114# 437# 549# 638# 706# 827# 880# 942#	114 437 549 638 706 827 880 942	187# 438# 565# 645# 707# 844# 881# 949#	187 438 565 645 707 844 881 949	
.ACM	70# 233 548 645# 712 858 949#	70 325# 549 645 719# 859 949	71 325 565# 646 719 880# 950	100# 326 565 653# 720 880	100 437# 566 653 826# 881	101 437 579# 654 826 918#	113# 438 579 683# 827 918	113 514# 580 683 844# 919	114 514 612# 684 844 933#	187# 515 612 706# 845 933	187 531# 613 706 851# 934	188 531 637# 707 851 941#	232# 532 637 711# 852 941	232 548# 638 711 858# 942	
.LBQRM	862														
.LBRMG	725														
.LBSMG	660														
.NVR	70# 188 514# 566 645 707# 844 881# 949	70 188# 514 566# 646 711# 845 918# 950	71 232# 515 579# 646# 711 845# 918 950#	71# 232 515# 579 653# 712 851# 919	100# 233 531# 580 653 712# 851 919#	100 233# 531 580# 654 719# 852 933#	101 325# 532 612# 654# 719 852# 933	101# 325 532# 612 683# 720 858# 934	113# 326 548# 613 683 720# 858 934#	113 326# 548 637# 684 826# 859 941#	114 437# 549 637 684# 826 859# 941	114# 437 549# 638 706# 827 880# 942	187# 438 565# 638# 706 827# 880 942#	187 438# 565 645# 707 844# 881 949#	
.POPJ	27														
.PSECT	70 645 949	100 653	113 683	187 706	232 711	325 719	437 826	514 844	531 851	548 858	565 880	579 918	612 933	637 941	
.RETF	27														
.RETT	27														
.SAVE2	612														
.SQAPT	667														
.SQARB	697														
.SQBID	735														
.SQCID	664														
.SQFLG	674														
.SQLEN	661														

.SSGDE	684
.SSQRD	852
.SSQRM	845
.SSRDG	712
.SSRMG	707
.SSSDG	646
.SSSMG	638

SEQ 0225

GLOB	27													
JUMPF	27													
JUMPT	27													
LSTOF.	26	71	96	101	114	180	183	188	228	233	315	318	321	326
	433	438	510	515	532	549	566	580	638	646	654	684	707	712
	720	827	845	852	859	881	889	896	904	907	919	934	942	950
LSTON.	71	101	114	188	233	326	438	515	532	549	566	580	613	638
	646	654	684	707	712	720	827	845	852	859	881	919	934	942
	950													
NAME	23													
PJRST	27													
PROLOG	26													
\$CALL	95	101	104	105	106	114	117	118	169	174	179	182	188	191
	192	211	221	227	233	236	237	279	290	295	304	309	314	317
	320	326	329	330	427	432	438	441	442	504	509	515	518	519
	526	532	535	536	543	549	552	553	560	566	569	570	574	580
	583	584	607	612	639	647	655	676	677	693	694	708	713	731
	732	809	846	853	872	873	888	895	903	906				
\$RET	27#	27	109	175	222	310	428	505	527	544	561	575	608	623
	629	640	648	679	699	709	714	813	837	847	854	875	907	929
	937	945	952											
\$RETF	27#													
\$RETIF	27#													
\$RETI	27#													
\$RETI	27#													
\$SAVE	70	100	113	187	232	325	437	514	531	548	565	579	612	637
	645	653	683	706	711	719	826	844	851	858	880	918	933	941
	949													
\$TEXT	95	179	182	227	314	317	320	432	509	886	895	903	906	
..POP	71	101	114	188	233	326	438	515	532	549	566	580	638	646
	654	684	707	712	720	827	845	852	859	881	919	934	942	950
..PUSH	71	101	114	188	233	326	438	515	532	549	566	580	638	646
	654	684	707	712	720	827	845	852	859	881	919	934	942	950
..TSAC	70	100	113	187	232	325	437	514	531	548	565	579	612	637
	645	653	683	706	711	719	826	844	851	858	880	918	933	941
	949													
.NV1	71#	71	101#	101	114#	114	188#	188	233#	233	326#	326	438#	438
	515#	515	532#	532	549#	549	566#	566	580#	580	638#	638	646#	646
	654#	654	684#	684	707#	707	712#	712	720#	720	827#	827	845#	845
	852#	852	859#	859	881#	881	919#	919	934#	934	942#	942	950#	950
.NV2	71#	71	101#	101	114#	114	188#	188	233#	233	326#	326	438#	438
	515#	515	532#	532	549#	549	566#	566	580#	580	654#	654	684#	684
	720#	720	827#	827	859#	859	881#	881	919#	919				
.NV3	71#	71	101#	101	114#	114	188#	188	233#	233	326#	326	438#	438
	515#	515	532#	532	549#	549	566#	566	580#	580	654#	654	684#	684
	720#	720	859#	859	881#	881								
.NV4	684#	684	720#	720										
.NV5	720#	720												

1
2
3
4
5
6
7
8
9
10

;Z:<GSCOTT.DFCIB>DFCIB3.MAC.16 14-Aug-85 16:15:10, Edit by GSCOTT
;Remove SCS JSYS calls, replace with calls to SCS\$.
;Z:<GSCOTT.DFCIB>DFCIB3.MAC.15 12-Aug-85 17:39:54, Edit by GSCOTT
;Correct externs
;Z:<GSCOTT.DFCIB>DFCIB3.MAC.13 12-Aug-85 17:26:02, Edit by GSCOTT
;Move DATSE, DATSR, RDATVM, RDAT20, RDATE, RDATE, SDATVM, SDAT20 from DFCIBP
;Z:<GSCOTT.DFCIB>DFCIB3.MAC.10 7-Aug-85 12:53:41, Edit by GSCOTT
;Remove GETD/PUTD
;Z:<GSCOTT.DFCIB>DFCIB3.MAC.7 6-Aug-85 20:05:06, Edit by GSCOTT
;Begin conversion to GLXLIB: title statement, ACs, etc.

```
11 SEARCH DFCIBT
12 NAME (DFCIB3,\DECVER,\VEDIT,Listen/Connect/Request Subroutines)^
13
14 SEARCH GLXMAC,MONSYM,UUOSYM,MACSYM
15
16 PROLOG (DFCIB3)^
17
18
19
20 SALL
21 .DIREC FLBLST
22 NOSYM
23
24 ;Interns
25
26 INTERN LISTEN,ACCEPT,LIRARG,ACCARG,RESPR
27 INTERN RDATE,RDATR,RDATVM,RDAT20,SDATVM,SDAT20,DATSE,DATSR
28
29 ;Externs in DFCIB1
30
31 EXTERN NAMED,NAMES,CONID,SCSERR,SCSERS,RETBUF,SCS$
32 EXTERN MBUF1,MBUF2,MBUF3,MBUF4,MBUF5,MBUF6
33 EXTERN DBUF1,DBUF2,DBUF3,DBUF4,DBUF5,DBUF6
34
35 ;Externs in DFCIB2
36
37 EXTERN CTPPKT,CTPLEN,SNMSG,SAVRSP,RLENBF,BUFLNM,BUFRNM,BFLNM1
38
39 ;Externs in DFCIBP
40
41 EXTERN NODEN
```

42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80

SUBTTL Listen

 ;This routine will listen for a responder in a system. Select the node
 ; by the "SELECT NODE x" command.
 ;First we set up the argument block then execute the scs jsys.
 ;Call:
 ; \$CALL Listen ;call this routine
 ; +1 ;Return here always
 ;This routine returns the connect id in location CONID .

LISTEN: \$SAVE	<S1,S2>	;Save the ac's
MOVEI	S1,LBLIS	;Get length
MOVEM	S1,LIRARG+.SQLEN	;Store length of argument block
MOVE	S2,[POINT 7,NAMED]	;Get byte pointer of "CTP\$RESPONDER"
MOVEM	S2,LIRARG+.SQSPN	;Store it for src process name
MOVE	S2,[POINT 7,NAMES]	;Get byte pointer of "CTP\$CONTROLLER"
MOVEM	S2,LIRARG+.SQDPN	;Store it for dst process name
HRLZI	S2,-1	;Listen to any node number
HRLI	S2,123450	;Get a SYSAP CID
MOVEM	S2,LIRARG+.SQSYS	;Store it in the argument block.
SETZM	LIRARG+.SQLCI	;Clear previous returned cid
MOVEI	S1,.SSLIS	;Listen function code
MOVEI	S2,LIRARG	;Address of argument block to s2
\$CALL	SCS\$;Perform SCS JSYS or UUD
\$CALL	SCSERR	;Output error message
MOVE	S1,LIRARG+.SQLCI	;Get the returned cid
MOVEM	S1,CONID	;Store it
\$RET		;Return to code that called this code

;Listen(.Sslis) for a responder argument block

LIRARG: 0...	LBLIS	;Processed words, length of block
0		;Byte ptr to source process name
0		;Byte ptr to destination process name
Z		;Node # of dest, sysap field for cid
Z		;Returned connect id(cid)


```

81          SUBTTL  Accept Request
82
83          ;*****
84          ;This routine tells remote node that we accept his request for a connection.
85          ;First we set up the argument block then execute the scs jsys.
86          ;Call:
87          ;      $call  accept          ;call this routine
88          ;      +1                ;Return here always
89          ;*****
90
91          000033' 304 00 0 00 000000  ACCEPT: $SAVE  <S1,S2>          ;Save the ac's
92          000040' 201 02 0 00 000003  MOVEI  S2,.LBACC          ;Get length of argument block
93          000041' 202 02 0 00 000056'  MOVEM  S2,ACCARG+.SQLEN      ;Store length of argument block
94          000042' 200 01 0 00 000024*  MOVE   S1,CONID          ;Get connect id
95          000043' 202 01 0 00 000057'  MOVEM  S1,ACCARG+.SQCID      ;Store the connect id
96          000044' 201 02 0 00 000061'  MOVEI  S2,CONDAT          ;Get connection data word address
97          000045' 202 02 0 00 000060'  MOVEM  S2,ACCARG+.SQCDA      ;Store it in the argument block.
98          000046' 201 01 0 00 000023  MOVEI  S1,.SSACC          ;Accept function code
99          000047' 201 02 0 00 000056'  MOVEI  S2,ACCARG          ;Address of argument block to s2
100         000050' 402 00 0 00 000000*  SETZM  SCSERS          ;Clear SCS failure indication
101         000051' 260 17 0 00 000021*  $CALL  SCS$          ;Perform SCS JSYS or ULO
102         000052' 260 17 0 00 000022*  $CALL  SCSERR          ;Output error message
103         000053' 336 00 0 00 000050*  SKIPN  SCSERS          ;Skip if a SCS error has occurred
104         000054' 260 17 0 00 000066'  $CALL  QLISB          ;Go queue buffers for listening
105         000055' 263 17 0 00 000000  $RET                ;Thats it
106
107          ;Accept(.Ssacc) argument block
108
109         000056' 000000 000003  ACCARG: 0,...LBACC          ;Processed words,,length of block
110         000057' 000 00 0 00 000000  Z          ;Connect id
111         000060' 000000 000061'  CONDAT          ;Ptr to connection data
112
113         000061'  CONDAT: BLOCK 5          ;Connection data
  
```

```

114                                SUBTTL Queue Listening Buffers
115
116                                ;*****
117                                ; Queue buffers for listening
118                                ;*****
119
120 000066' 304 00 0 00 000000    QLISB: $SAVE    <S2,S1>                ;Save ACs
121 000073' 201 01 0 00 000000*    MOVEI    S1,MBUF2            ;Get address of 1st message buffer
122 000074' 202 01 0 00 000000*    MOVEM    S1,MBUF1            ;Link 1st buffer to second buffer
123 000075' 201 01 0 00 000000*    MOVEI    S1,MBUF3            ;Get address of 3rd message buffer
124 000076' 202 01 0 00 000073*    MOVEM    S1,MBUF2            ;Link 2nd buffer to 3rd buffer
125 000077' 201 01 0 00 000000*    MOVEI    S1,MBUF4            ;Get address of 4th message buffer
126 000100' 202 01 0 00 000075*    MOVEM    S1,MBUF3            ;Link 4th buffer to 3rd buffer
127 000101' 201 01 0 00 000000*    MOVEI    S1,MBUF5            ;Get address of 5th message buffer
128 000102' 202 01 0 00 000077*    MOVEM    S1,MBUF4            ;Link 5th buffer to 4th buffer
129 000103' 201 01 0 00 000000*    MOVEI    S1,MBUF6            ;Get address of 6th message buffer
130 000104' 202 01 0 00 000101*    MOVEM    S1,MBUF5            ;Link 6th buffer to 5th buffer
131 000105' 402 00 0 00 000103*    SETZM    MBUF6                ;Mark end of message buffer chain
132
133 000106' 201 01 0 00 000000*    MOVEI    S1,DBUF2            ;Get address of 1st datagram buffer
134 000107' 202 01 0 00 000000*    MOVEM    S1,DBUF1            ;Link 1st buffer to second buffer
135 000110' 201 01 0 00 000000*    MOVEI    S1,DBUF3            ;Get address of 3rd datagram buffer
136 000111' 202 01 0 00 000106*    MOVEM    S1,DBUF2            ;Link 2nd buffer to 3rd buffer
137 000112' 201 01 0 00 000000*    MOVEI    S1,DBUF4            ;Get address of 4th datagram buffer
138 000113' 202 01 0 00 000110*    MOVEM    S1,DBUF3            ;Link 4th buffer to 3rd buffer
139 000114' 201 01 0 00 000000*    MOVEI    S1,DBUF5            ;Get address of 5th datagram buffer
140 000115' 202 01 0 00 000112*    MOVEM    S1,DBUF4            ;Link 5th buffer to 4th buffer
141 000116' 201 01 0 00 000000*    MOVEI    S1,DBUF6            ;Get address of 6th datagram buffer
142 000117' 202 01 0 00 000114*    MOVEM    S1,DBUF5            ;Link 6th buffer to 5th buffer
143 000120' 402 00 0 00 000116*    SETZM    DBUF6                ;Mark end of datagram buffer chain
144
145 000121' 201 02 0 00 000074*    MOVEI    S2,MBUF1            ;Put 1st message buffer address in S2
146 000122' 201 01 0 00 000007*    MOVEI    S1,SSQRM            ;Put queue message buffer in S1
147 000123' 260 17 0 00 000130'    $CALL    LQBUF                ;Queue message buffers
148
149 000124' 201 02 0 00 000107*    MOVEI    S2,DBUF1            ;Put 1st datagram buffer address in S2
150 000125' 201 01 0 00 000005*    MOVEI    S1,SSQRD            ;Put queue datagram buffer in S1
151 000126' 260 17 0 00 000130'    $CALL    LQBUF                ;Queue datagram buffers
152 000127' 263 17 0 00 000000*    $RET

```

```

153
154
155 ;*****
156 ; Call with either .SSQRD (queue datagram buffer) or .SSQRM (queue message
157 ; buffer) in S1, and the start address of the message or datagram buffer
158 ; chain in ACO.
159 ;*****
160 000130' 202 02 0 00 000143' LQBUF: MOVEM S2,LQBUFB+.SQAFB ;Store start address of buffer chain
161 000131' 201 02 0 00 000003' MOVEI S2,.LBQRD ;Get length of argument block
162 000132' 202 02 0 00 000141' MOVEM S2,LQBUFB+.SQLEN ;Store it
163 000133' 200 02 0 00 000042* MOVE S2,CONID ;Get connection id
164 000134' 202 02 0 00 000142' MOVEM S2,LQBUFB+.SQCID ;Store connect id
165
166 000135' 201 02 0 00 000141' MOVEI S2,LQBUFB ;Argument block address
167 000136' 260 17 0 00 000051* $CALL SC$S ;Perform SCS JSYS or UUO
168 000137' 260 17 0 00 000052* $CALL SCSERR ;Output error message
169 000140' 263 17 0 00 000000 $RET ;Return to calling
170
171 000141' 000000 000003 LQBUFB: 0,..LBQRD ;Processed words (mon returns),,
172 ; Length of block (user supplied)
173 000142' 000000 000000 0 ;Connection id
174 000143' 000000 000000 0 ;Start address of buffer chain

```

```

175          SUBTTL Return Message
176
177          ;*****
178          ;Called via command 'RETURN-MESSAGE'
179          ;*****
180
181 000144' 304 00 0 00 000000 RESPR: $SAVE <S1> ;Save ACs
182 000150' 135 01 0 00 000531' LDB S1,[POINT 8,@RETBUF,7] ;Get opcode
183 000151' 303 01 0 00 000015 CAILE S1,^D13 ;Skip if opcode .LE. 13.
184 000152' 201 01 0 00 000005 MOVEI S1,5 ;Force not implemented
185 000153' 260 17 1 01 000155' $CALL @OPTABL(S1) ;Do the opcode
186 000154' 263 17 0 00 000000 $RET
187
188 000155' 000000 000173' OPTABL: R0
189 000156' 000000 000212' RNIMP ;1
190 000157' 000000 000212' RNIMP ;2
191 000160' 000000 000212' RNIMP ;3
192 000161' 000000 000212' RNIMP ;4
193 000162' 000000 000212' RNIMP ;5
194 000163' 000000 000212' RNIMP ;6
195 000164' 000000 000212' RNIMP ;7
196 000165' 000000 000212' RNIMP ;8
197 000166' 000000 000212' RNIMP ;9
198 000167' 000000 000212' RNIMP ;10
199 000170' 000000 000212' RNIMP ;11
200 000171' 000000 000212' RNIMP ;12
201 000172' 000000 000212' RNIMP ;13

```



```

202
203 000173' 304 00 0 00 000000      R0:  $SAVE  <S1>           ;Save an AC
204 000177' 260 17 0 00 000226'      $CALL  RCOPYB        ;Copy SAVRSP to CTPPKT
205 000200' 260 17 0 00 000250'      $CALL  RMOPC         ;Change the OPCODE field
206 000201' 201 01 0 00 000000      MOVEI  S1,^D0          ;Get success status
207 000202' 137 01 0 00 000532'      DPB      S1,[POINT 8,CTPPKT+1,^D16] ;Write the status byte
208 000203' 201 01 0 00 174174'      MOVEI  S1,174174        ;Get implemented opcodes 0-13
209 000204' 137 01 0 00 000533'      DPB      S1,[POINT ^D16,CTPPKT+1,^D32] ;Write it in 0-15 field
210 000205' 137 01 0 00 000534'      DPB      S1,[POINT ^D16,CTPPKT+3,^D32] ;Write it in 64-79 field
211 000206' 201 01 0 00 000046'      MOVEI  S1,^D38          ;Byte length = 38
212 000207' 202 01 0 00 000000*      MOVEM  S1,CTPLEN      ;Put the length away
213 000210' 260 17 0 00 000000*      $CALL  SNDMSG        ;Send the message back
214 000211' 263 17 0 00 000000      $RET          ;Return to calling code
215
216 000212' 304 00 0 00 000000      RNIMP: $SAVE  <S1>           ;Save AC
217 000216' 260 17 0 00 000226'      $CALL  RCOPYB        ;Copy SAVRSP to CTPPKT
218 000217' 260 17 0 00 000250'      $CALL  RMOPC         ;Change the OPCODE field
219 000220' 201 01 0 00 000377'      MOVEI  S1,^D255        ;Function not implemented stat
220 000221' 137 01 0 00 000535'      DPB      S1,[POINT 8,CTPPKT+1,^D16] ;Write the status byte
221 000222' 200 01 0 00 000000*      MOVE   S1,RLENBF      ;Get read byte length
222 000223' 202 01 0 00 000207*      MOVEM  S1,CTPLEN      ;Put the length away
223 000224' 260 17 0 00 000210*      $CALL  SNDMSG        ;Send the message back
224 000225' 263 17 0 00 000000      $RET          ;Return to calling code
225
226 000226' 304 00 0 00 000000      RCOPYB: $SAVE  <T2,T1,S2,S1>      ;Save some ACs
227
228 000235' 402 00 0 00 000000*      SETZM  CTPPKT        ;Zero first word
229 000236' 200 01 0 00 000546'      MOVE   S1,[CTPPKT,,CTPPKT+1] ;Get BLT word
230 000237' 251 01 0 00 000000#      BLT    S1,CTPPKT+46  ;Zero CTPPKT buffer
231
232 000240' 201 03 0 00 000222*      MOVE   T1,RLENBF      ;Get received byte length
233 000241' 200 02 0 00 000547'      MOVE   S2,[POINT 8,0]  ;Get initial pointer to source
234 000242' 270 02 0 00 000000*      ADD    S2,RETBUF      ;Add in M buf address
235 000243' 200 01 0 00 000550'      MOVE   S1,[POINT 8,CTPPKT] ;Get pointer to destination
236 000244' 134 04 0 00 000002      RCPYB1: ILDB  S2,S1      ;Get source byte
237 000245' 136 04 0 00 000001      IDPB  S1,S1      ;Deposit the byte
238 000246' 366 02 0 00 000244'      SOJN   S1,RCPYB1      ;Loop back until all copied
239 000247' 263 17 0 00 000000      $RET          ;Return to calling code
240
241      ;Change the OPCODE field. It's assumed that the buffer has been copied to
242      ;CTPPKT.
243
244 000250' 304 00 0 00 000000      RMOPC: $SAVE  <S1>           ;Save an AC
245 000254' 135 01 0 00 000551'      LDB     S1,[POINT 8,CTPPKT,7] ;Get OPCODE field
246 000255' 271 01 0 00 000100'      ADDI   S1,^D64        ;Modify to response opcode
247 000256' 137 01 0 00 000551'      DPB      S1,[POINT 8,CTPPKT,7] ;Write it back
248 000257' 263 17 0 00 000000      $RET          ;Return to calling code

```



```

249          SUBTTL Request Data
250
251          ;Request data from responder-node - TOPS20
252
253          RDAT20: $CALL DATC1          ;See if buffers were mapped
254                  $RET                ;They weren't
255                  $SAVE <S2,S1>       ;Save ACs
256                  MOVE S1,BFLNM1      ;Get respndr's 10 name.
257                  MOVEM S1,DATAB+.SQSNM ;Put it in SCS block
258                  MOVE S1,BUFRNM      ;Get exerciser's buffer name
259                  MOVEM S1,DATAB+.SQRNM ;Put it in SCS block
260                  SETZM DATAB+.SQOFS   ;Use 0 offset
261                  MOVEI S1,.SSREQ      ;Put request data command in AC
262                  $CALL DATAO         ;Do the command
263                  $RET                ;Return to mainline
264
265          ;Request data from responder-node- USING VMS
266
267          RDATVM: $CALL DATC1          ;See if buffers were mapped
268                  $RET                ;They weren't
269                  $SAVE <S2,S1>       ;Save ACs
270                  MOVE S1,BUFLNM      ;Get responder's buffer name
271                  MOVEM S1,DATAB+.SQSNM ;Put it in SCS block
272                  MOVE S1,BUFRNM      ;Get exerciser's buffer name
273                  MOVEM S1,DATAB+.SQRNM ;Put it in SCS block
274                  SETZM DATAB+.SQOFS   ;Use 0 offset
275                  MOVEI S1,.SSREQ      ;Put request data command in AC
276                  $CALL DATAO         ;Do the command
277                  $RET                ;Return to mainline

```

```

278
279      ;Request data from responder-node (REQUEST RESPONDER command)
280
281 000316' 260 17 0 00 000450'  RDATR: $CALL DATC1      ;See if buffers were mapped
282 000317' 263 17 0 00 000000'    $RET      ;They weren't
283 000320' 304 00 0 00 000000'    $SAVE <S2,S1> ;Save ACs
284 000325' 200 01 0 00 000267*   MOVE S1,BFLNM1 ; GET RESPNDR'S 10 NAME.
285 000326' 202 01 0 00 000511'   MOVEM S1,DATAB+.SQSNM ;Put it in SCS block
286 000327' 200 01 0 00 000310*   MOVE S1,BUFRNM ;Get exerciser's buffer name
287 000330' 202 01 0 00 000512'   MOVEM S1,DATAB+.SQRNM ;Put it in SCS block
288 000331' 402 00 0 00 000513'   SETZM DATAB+.SQOFS ;Use 0 offset
289 000332' 201 01 0 00 000017'   MOVEI S1,.SSREQ ;Put request data command in AC
290 000333' 260 17 0 00 000472'   $CALL DATAO ;Do the command
291 000334' 263 17 0 00 000000'    $RET      ;Return to mainline
292
293      ;Request data from exerciser-node
294
295 000335' 260 17 0 00 000450'  RDATE: $CALL DATC1      ;See if buffers were mapped
296 000336' 263 17 0 00 000000'    $RET      ;They weren't
297 000337' 304 00 0 00 000000'    $SAVE <S2,S1> ;Save ACs
298 000344' 200 01 0 00 000327*   MOVE S1,BUFRNM ;Get exerciser's buffer name
299 000345' 202 01 0 00 000511'   MOVEM S1,DATAB+.SQSNM ;Put it in SCS block
300 000346' 200 01 0 00 000306*   MOVE S1,BUFLNM ;Get responder's buffer name
301 000347' 202 01 0 00 000512'   MOVEM S1,DATAB+.SQRNM ;Put it in SCS block
302 000350' 402 00 0 00 000513'   SETZM DATAB+.SQOFS ;Use 0 offset
303 000351' 201 01 0 00 000017'   MOVEI S1,.SSREQ ;Put request data command in AC
304 000352' 260 17 0 00 000472'   $CALL DATAO ;Do the command
305 000353' 263 17 0 00 000000'    $RET      ;Return to mainline
  
```

```

306
307 ;Send data to responder-node TOPS20.
308
309 000354' 260 17 0 00 000450' SDAT20: $CALL DATC1 ;See if buffers were mapped
310 000355' 263 17 0 00 000000 $RET ;They weren't
311 000356' 304 00 0 00 000000 $SAVE <S2,S1> ;Save ACs
312 000363' 200 01 0 00 000344* MOVE S1,BUFRNM ;Get exerciser's buffer name
313 000364' 202 01 0 00 000511' MOVEM S1,DATAB+.SQSNM ;Put it in SCS block
314 000365' 200 01 0 00 000325* MOVE S1,BFLNM1 ;Get TOPS20 responder's buffer name
315 000366' 202 01 0 00 000512' MOVEM S1,DATAB+.SQRNM ;Put it in SCS block
316 000367' 402 00 0 00 000513' SETZM DATAB+.SQOFS ;Use 0 offset
317 000370' 201 01 0 00 000016 MOVEI S1,SSSND ;Put send data command in AC
318 000371' 260 17 0 00 000472' $CALL DATAO ;Do the command
319 000372' 263 17 0 00 000000 $RET ;Return to mainline
320
321 ;Send data to responder-node with VMS running
322
323 000373' 260 17 0 00 000450' SDATVM: $CALL DATC1 ;See if buffers were mapped
324 000374' 263 17 0 00 000000 $RET ;They weren't
325 000375' 304 00 0 00 000000 $SAVE <S2,S1> ;Save ACs
326 000402' 200 01 0 00 000363* MOVE S1,BUFRNM ;Get exerciser's buffer name
327 000403' 202 01 0 00 000511' MOVEM S1,DATAB+.SQSNM ;Put it in SCS block
328 000404' 200 01 0 00 000346* MOVE S1,BUFLNM ;Get VMS responder's buffer name
329 000405' 202 01 0 00 000512' MOVEM S1,DATAB+.SQRNM ;Put it in SCS block
330 000406' 402 00 0 00 000513' SETZM DATAB+.SQOFS ;Use 0 offset
331 000407' 201 01 0 00 000016 MOVEI S1,SSSND ;Put send data command in AC
332 000410' 260 17 0 00 000472' $CALL DATAO ;Do the command
333 000411' 263 17 0 00 000000 $RET ;Return to mainline

```

```

334
335 ;Send data to responder-node
336
337 000412' 260 17 0 00 000450' DATSR: $CALL DATC1 ;See if buffers were mapped
338 000413' 263 17 0 00 000000 $RET ;They weren't
339 000414' 304 00 0 00 000000 $SAVE <S2,S1> ;Save ACs
340 000421' 200 01 0 00 000402* MOVE S1,BUFRNM ;Get exerciser's buffer name
341 000422' 202 01 0 00 000511' MOVEM S1,DATAB+.SQSNM ;Put it in SCS block
342 000423' 200 01 0 00 000404* MOVE S1,BUFLNM ;Get responder's buffer name
343 000424' 202 01 0 00 000512' MOVEM S1,DATAB+.SQRNM ;Put it in SCS block
344 000425' 402 00 0 00 000513' SETZM DATAB+.SQOFS ;Use 0 offset
345 000426' 201 01 0 00 000016 MOVEI S1,.SSSND ;Put send data command in AC
346 000427' 260 17 0 00 000472' $CALL DATAO ;Do the command
347 000430' 263 17 0 00 000000 $RET ;Return to mainline
348
349 ;Send data to exerciser-node
350
351 000431' 260 17 0 00 000450' DATSE: $CALL DATC1 ;See if buffers were mapped
352 000432' 263 17 0 00 000000 $RET ;They weren't
353 000433' 304 00 0 00 000000 $SAVE <S2,S1> ;Save ACs
354 000440' 200 01 0 00 000423* MOVE S1,BUFLNM ;Get responder's buffer name
355 000441' 202 01 0 00 000511' MOVEM S1,DATAB+.SQSNM ;Put it in SCS block
356 000442' 200 01 0 00 000421* MOVE S1,BUFRNM ;Get exerciser's buffer name
357 000443' 202 01 0 00 000512' MOVEM S1,DATAB+.SQRNM ;Put it in SCS block
358 000444' 402 00 0 00 000513' SETZM DATAB+.SQOFS ;Use 0 offset
359 000445' 201 01 0 00 000016 MOVEI S1,.SSSND ;Put send data command in AC
360 000446' 260 17 0 00 000472' $CALL DATAO ;Do the command
361 000447' 263 17 0 00 000000 $RET ;Return to mainline

```



```

362
363 ;Check if buffers are mapped. If not return +1. Otherwise return +2.
364
365 000450' 304 00 0 00 000000 DATC1: $SAVE <S1> ;Save an AC
366 000454' 402 00 0 00 000001 SETZM S1 ;Zero S1
367 000455' 332 00 0 00 000440* SKIPE BUFLNM ;Skip if responder not mapped
368 000456' 254 00 0 00 000462' JRST DATC1A ;It was mapped
369 000457' 350 00 0 00 000001 AOS S1 ;It wasn't
370 000460' 260 17 0 00 000000* $TEXT (,<? Responder buffer was not mapped>)
371 000462' 332 00 0 00 000442* DATC1A: SKIPE BUFRNM ;Skip if exerciser not mapped
372 000463' 254 00 0 00 000467' JRST DATC1B ;It was mapped
373 000464' 350 00 0 00 000001 AOS S1 ;It wasn't
374 000465' 260 17 0 00 000460* $TEXT (,<? Exerciser buffer was not mapped>)
375 000467' 336 00 0 00 000001 DATC1B: SKIPN S1 ;Are they mapped (S1/0) ?
376 000470' 350 00 0 17 000000 AOS (P) ;Yes
377 000471' 263 17 0 00 000000 $RET ;Return to code that called this code
378
379 ;Do the SEND DATA or RECEIVE DATA command. Call with the command, either .SSSND
380 ; or .SSREQ in S1. It's assumed that locations .SQSNM, .SQRNM and .SQOFS are
381 ; set up prior to this subroutine.
382
383 000472' 304 00 0 00 000000 DATA0: $SAVE <S2,S1> ;Save some ACs
384 000477' 201 02 0 00 000005 MOVEI S2,.LBSND ;Get length of argument block
385 000500' 202 02 0 00 000507' MOVEM S2,DATAB+.SQLEN ;Put it in argument block
386 000501' 200 02 0 00 000133* MOVE S2,CONID ;Get the connect id
387 000502' 202 02 0 00 000510' MOVEM S2,DATAB+.SQCID ;Put the connect id in argument block
388 000503' 201 02 0 00 000507' MOVEI S2,DATAB ;Get length of argument block
389 000504' 260 17 0 00 000136* $CALL SC$S ;Perform SCS JSYS or UUO
390 000505' 260 17 0 00 000137* $CALL SCSERR ;Output error message
391 000506' 263 17 0 00 000000 $RET
392
393 000507' 000000 000000 DATAB: 0 ;Length of block
394 000510' 000000 000000 0 ;CID
395 000511' 000000 000000 0 ;Buffer name of send buffer
396 000512' 000000 000000 0 ;Buffer name of receive buffer
397 000513' 000000 000000 0,,0 ;Xmit offset,,Receive offset

```



```
398                                SUBTTL Literals
399
400 000514'                       LIT..3: XLIST           ;Lit
401                                LIST
402
403                                END
```

NO ERRORS DETECTED

PROGRAM BREAK IS 000617
CPU TIME USED 00:10.359

148P CORE USED

ACCARG	26	93	95	97	99	109#								
ACCEPT	26	91#												
BFLNM1	37#	256	284	314										
BUFLNM	37#	270	300	328	342	354	367							
BUFRNM	37#	258	272	286	298	312	326	340	356	371				
CONDAT	96	111	113#											
CONID	31#	70	94	163	386									
CTPLEN	37#	212	222											
CTPPKT	37#	207	209	210	220	228	229	230	235	245	247			
DATAB	257	259	260	271	273	274	285	287	288	299	301	302	313	315
	316	327	329	330	341	343	344	355	357	358	385	387	388	393#
DATAO	262	276	290	304	318	332	346	360	383#					
DATC1	253	267	281	295	309	323	337	351	365#					
DATC1A	368	371#												
DATC1B	372	375#												
DATSE	27	351#												
DATSR	27	337#												
DBUF1	33#	134	149											
DBUF2	33#	133	136											
DBUF3	33#	135	138											
DBUF4	33#	137	140											
DBUF5	33#	139	142											
DBUF6	33#	141	143											
DECVER	13													
LIRARG	26	56	58	60	63	64	66	69	76#					
LISTEN	26	54#												
LIT...3	400#													
LQBUF	147	151	160#											
LQBUFB	160	162	164	166	171#									
LSTIN.	18	55	55#	92	92#	121	121#	182	182#	204	204#	217	217#	227
	227#	245	245#	256	256#	270	270#	284	284#	298	298#	312	312#	326
	326#	340	340#	354	354#	366	366#	371	375	384	384#			
MBUF1	32#	122	145											
MBUF2	32#	121	124											
MBUF3	32#	123	126											
MBUF4	32#	125	128											
MBUF5	32#	127	130											
MBUF6	32#	129	131											
NAMED	31#	57												
NAMES	31#	59												
NODEN	41#													
OPTABL	185	188#												
P	18	55	92	121	182	204	217	227	245	256	270	284	298	312
	326	340	354	366	376	384								
P1	54	91	120	181	203	216	226	244	255	269	283	297	311	325
	339	353	365	383										

R0	188	203#												
RCOPYB	204	217	226#											
RCPYB1	236#	238												
RDAT20	27	253#												
RDATE	27	295#												
RDATR	27	281#												
RDATVM	27	267#												
RESPR	26	181#												
RETBUF	31#	182	234											
RLENBF	37#	221	232											
RMOPC	205	218	244#											
RNIMP	189	190	191	192	193	194	195	196	197	198	199	200	201	216#
S1	54	55	56	65	69	70	91	94	95	98	120	121	122	123
	124	125	126	127	128	129	130	133	134	135	136	137	138	139
	140	141	142	146	150	181	182	183	184	185	203	206	207	208
	209	210	211	212	216	219	220	221	222	226	229	230	235	237
	244	245	246	247	255	256	257	258	259	261	269	270	271	272
	273	275	283	284	285	286	287	289	297	298	299	300	301	303
	311	312	313	314	315	317	325	326	327	328	329	331	339	340
	341	342	343	345	353	354	355	356	357	359	365	366	369	373
	375	383												
S2	54	57	58	59	60	61	62	63	66	91	92	93	96	97
	99	120	145	149	160	161	162	163	164	166	226	233	234	236
	255	269	283	297	311	325	339	353	383	384	385	386	387	388
SAVRSP	37#													
SCS\$	31#	67	101	167	389									
SCSERR	31#	68	102	168	390									
SCSERS	31#	100	103											
SDAT20	27	309#												
SDATVM	27	323#												
SNDMSG	37#	213	223											
TXTTEXT	370	374												
T1	226	232	238											
T2	226	236	237											
T4	54	91	120	181	203	216	226	244	255	269	283	297	311	325
	339	353	365	383										
VEDIT	13													
\$RETF	18													
\$RETIF	18													
\$RETIT	18													
\$RETT	18													
..0001	55	55#												
..0002	92	92#												
..0003	121	121#												
..0004	182	182#												
..0005	204	204#												
..0006	217	217#												
..0007	227	227#												
..0010	245	245#												
..0011	256	256#												
..0012	270	270#												
..0013	284	284#												
..0014	298	298#												

.SSACC	98			
.SSLIS	65			
.SSQRD	150			
.SSQRM	146			
.SSREQ	261	275	289	303
.SSSND	317	331	345	359

[illegible]

1	:Z:<GSCOTT.DFCIB>DFCIBP.MAC.278 27-Aug-85 17:31:12, Edit by GSCOTT
2	:Fix bug in DDT command on 20: didn't go to preloaded UDDT.
3	:Z:<GSCOTT.DFCIB>DFCIBP.MAC.276 26-Aug-85 13:52:29, Edit by GSCOTT
4	:Unlock job when escape typed at runtime to abort testing.
5	:Z:<GSCOTT.DFCIB>DFCIBP.MAC.270 23-Aug-85 10:34:47, Edit by GSCOTT
6	:Allow "T" to toggle trace mode at run time
7	:Z:<GSCOTT.DFCIB>DFCIBP.MAC.258 22-Aug-85 22:57:40, Edit by GSCOTT
8	:Add routine RTCHK to output current run time status.
9	:Z:<GSCOTT.DFCIB>DFCIBP.MAC.256 22-Aug-85 19:41:13, Edit by GSCOTT
10	:Change SHOW CONNECTION-DATA-OF-ALL-NODES to SHOW CI-CONFIGURATION
11	:Z:<GSCOTT.DFCIB>DFCIBP.MAC.246 22-Aug-85 13:46:23, Edit by GSCOTT
12	:Improve DDT command
13	:Z:<GSCOTT.DFCIB>DFCIBP.MAC.243 22-Aug-85 13:12:33, Edit by GSCOTT
14	:UNLKIT never unlocked because it was checking the flag wrong.
15	:Z:<GSCOTT.DFCIB>DFCIBP.MAC.241 21-Aug-85 19:06:35, Edit by GSCOTT
16	:Needed to call CMDLOG in PARERR
17	:Z:<GSCOTT.DFCIB>DFCIBP.MAC.240 21-Aug-85 18:59:50, Edit by GSCOTT
18	:Add output of switches to SHOW ALL
19	:Z:<GSCOTT.DFCIB>DFCIBP.MAC.239 21-Aug-85 15:52:42, Edit by GSCOTT
20	:Change SHOW ALL, add SHOW PATH-SELECTION command
21	:Z:<GSCOTT.DFCIB>DFCIBP.MAC.234 21-Aug-85 14:37:22, Edit by GSCOTT
22	:Spelled RESPONSE badly in WAITING-FOR-DISCONNECT-RESPONSE
23	:Z:<GSCOTT.DFCIB>DFCIBP.MAC.230 21-Aug-85 12:26:00, Edit by GSCOTT
24	:Fix parsing of SELECT and DESELECT commands.
25	:Z:<GSCOTT.DFCIB>DFCIBP.MAC.226 21-Aug-85 12:00:25, Edit by GSCOTT
26	:Change SELECT NODE and DESELECT NODE to drop the "NODE" part.
27	:Z:<GSCOTT.DFCIB>DFCIBP.MAC.222 20-Aug-85 20:46:23, Edit by GSCOTT
28	:BGENC broken, HLPARS pointed to a loop.
29	:Z:<GSCOTT.DFCIB>DFCIBP.MAC.219 20-Aug-85 17:09:21, Edit by GSCOTT
30	:OUTSTR a "C" if the control C was typed at a prompt
31	:Z:<GSCOTT.DFCIB>DFCIBP.MAC.216 20-Aug-85 16:20:22, Edit by GSCOTT
32	:Set a ^C trap on the 10 to unlock ourselves before exiting
33	:Z:<GSCOTT.DFCIB>DFCIBP.MAC.213 20-Aug-85 15:21:03, Edit by GSCOTT
34	:Be sure to unlock ourselves if HALT switch set. Remove ITRFLG
35	:Z:<GSCOTT.DFCIB>DFCIBP.MAC.209 20-Aug-85 12:12:07, Edit by GSCOTT
36	:Clean up command table
37	:Z:<GSCOTT.DFCIB>DFCIBP.MAC.204 16-Aug-85 10:07:38, Edit by GSCOTT
38	:Implement BELL switch
39	:Z:<GSCOTT.DFCIB>DFCIBP.MAC.202 15-Aug-85 23:58:29, Edit by GSCOTT
40	:Move impure storage to try and catch stray store bug
41	:Z:<GSCOTT.DFCIB>DFCIBP.MAC.200 15-Aug-85 21:35:13, Edit by GSCOTT
42	:Fix SHOW SELECTED-NODES (there are 16 nodes not 15)
43	:Z:<GSCOTT.DFCIB>DFCIBP.MAC.198 15-Aug-85 14:59:02, Edit by GSCOTT
44	:Fix LISTEN/ACCEPT commands to parse confirm
45	:Z:<GSCOTT.DFCIB>DFCIBP.MAC.197 15-Aug-85 14:53:03, Edit by GSCOTT
46	:Lock in core on TOPS-10 when running tests.
47	:Z:<GSCOTT.DFCIB>DFCIBP.MAC.196 15-Aug-85 12:50:33, Edit by GSCOTT
48	:Print version from JBVER
49	:Z:<GSCOTT.DFCIB>DFCIBP.MAC.192 13-Aug-85 14:25:19, Edit by GSCOTT
50	:Implement HALT switch in ERRHAN.
51	:Z:<GSCOTT.DFCIB>DFCIBP.MAC.190 13-Aug-85 13:52:06, Edit by GSCOTT
52	:Intern ITCNT for printing from exerciser
53	:Z:<GSCOTT.DFCIB>DFCIBP.MAC.189 13-Aug-85 13:47:21, Edit by GSCOTT
54	:Rework pass and iteration counting logic.
55	:Z:<GSCOTT.DFCIB>DFCIBP.MAC.184 13-Aug-85 11:41:11, Edit by GSCOTT

```

56      ;SET TIME-OUT didn't parse the number
57      ;Z:<GSCOTT.DFCIB>DFCIBP.MAC.181 12-Aug-85 22:40:54, Edit by GSCOTT
58      ;Default iteration count properly here rather than resorting to test 99 hack
59      ;Z:<GSCOTT.DFCIB>DFCIBP.MAC.179 12-Aug-85 22:35:40, Edit by GSCOTT
60      ;Move SCRTAB and TSTTAB here from DFNISM
61      ;Z:<GSCOTT.DFCIB>DFCIBP.MAC.176 12-Aug-85 17:27:03, Edit by GSCOTT
62      ;Move ERRHAN from DFCIBM
63      ;Z:<GSCOTT.DFCIB>DFCIBP.MAC.175 12-Aug-85 17:26:15, Edit by GSCOTT
64      ;Move DATSE, DATSR, RDATVM, RDAT20, RDATE, RDATE, SDATVM, SDAT20 to DFCIB3
65      ;Z:<GSCOTT.DFCIB>DFCIBP.MAC.173 12-Aug-85 13:54:18, Edit by GSCOTT
66      ;Repair SHOW SELECT
67      ;Z:<GSCOTT.DFCIB>DFCIBP.MAC.172 12-Aug-85 12:43:55, Edit by GSCOTT
68      ;Default SELECT keyword to NODE
69      ;Z:<GSCOTT.DFCIB>DFCIBP.MAC.167 12-Aug-85 12:27:10, Edit by GSCOTT
70      ;Remove unreferenced cell MCBFLG
71      ;Z:<GSCOTT.DFCIB>DFCIBP.MAC.165 3-Aug-85 23:34:48, Edit by GSCOTT
72      ;Fix bad prompt if in take file
73      ;Z:<GSCOTT.DFCIB>DFCIBP.MAC.163 8-Aug-85 23:10:05, Edit by GSCOTT
74      ;Delete ITCNTI, remove intern of ITCNT
75      ;Z:<GSCOTT.DFCIB>DFCIBP.MAC.148 8-Aug-85 19:59:51, Edit by GSCOTT
76      ;Fix SET/CLEAR SWITCHES, SHOW SELECTED-NODES, SHOW MAPPED-BUFFER-NAME
77      ;Z:<GSCOTT.DFCIB>DFCIBP.MAC.138 8-Aug-85 19:03:35, Edit by GSCOTT
78      ;Somehow I broke the RUN command. Revamp it and add comments.
79      ;Z:<GSCOTT.DFCIB>DFCIBP.MAC.125 8-Aug-85 15:51:58, Edit by GSCOTT
80      ;Output scripts before tests in parse table.
81      ;Z:<GSCOTT.DFCIB>DFCIBP.MAC.124 8-Aug-85 15:42:21, Edit by GSCOTT
82      ;Move SCRTAB to DFCIBM.
83      ;Z:<GSCOTT.DFCIB>DFCIBP.MAC.122 8-Aug-85 14:52:21, Edit by GSCOTT
84      ;Intern PCRLF
85      ;Z:<GSCOTT.DFCIB>DFCIBP.MAC.112 7-Aug-85 14:58:23, Edit by GSCOTT
86      ;Use MAPBBA (page from GLXLIB) for address of MAPBB.
87      ;Z:<GSCOTT.DFCIB>DFCIBP.MAC.109 7-Aug-85 13:01:41, Edit by GSCOTT
88      ;Remove GETD/PUTD, call to SETDP
89      ;Z:<GSCOTT.DFCIB>DFCIBP.MAC.99 6-Aug-85 20:19:12, Edit by GSCOTT
90      ;Remove UTLTAB
91      ;Z:<GSCOTT.DFCIB>DFCIBP.MAC.69 6-Aug-85 11:03:21, Edit by GSCOTT
92      ;Add parsing subroutines and logging subroutines.
93      ;Z:<GSCOTT.DFCIB>DFCIBP.MAC.55 5-Aug-85 16:20:34, Edit by GSCOTT
94      ;Add MONTYP, code to support setting it in SETUP.
95      ;Z:<GSCOTT.DFCIB>DFCIBP.MAC.46 5-Aug-85 14:35:04, Edit by GSCOTT
96      ;Start conversion to GLXLIB, change AC names, remove all occurrences of ACO,
97      ;use $TEXT macro, use GLXLIB parsing, insert new help file reader, rewrite
98      ;SET/CLEAR/SHOW SWITCH commands.
99
100     ; 5/8/85 REMOVE EXTERNAL REFERENCE TO WAITC,WAITE. REMOVE CODE REFERENCE
101     ; TO WAITC IN .WCS1,.WCS2. REMOVE CODE REFERENCE TO WAITE IN .WES.
102

```

```
103
104 SEARCH DFCIBT
105 NAME (DFCIBP,\DECVER,\VEDIT,Parse and Dispatch)^
106
107 SEARCH GLXMAC,MONSYM,UUOSYM,MACSYM
108
109 PROLOG (DFCIBP)^
110
111
112 SALL
113 .DIREC FLBLST
114 NOSYM
115
116 ;Interns for routines/data found here in DFCIBP
117
118 INTERN CHKEV,CHKST,SECS,MONTYP,PCRLF,SNODET,ERRHAN,RTCHK
119 INTERN PASSCN,PASCNT,TITCNT,ITCNT,SWTRAC,TSTNAM,TSTDSC
120 INTERN NODEN,MAPBBA,PATHS,GETFN,GETNN,DESNA,SELNA
121
122 ;Externs found in DFCIB1
123
124 EXTERN DBUF1,DBUF2,DBUF3,MBUF1,MBUF2,MBUF3,SCSERS,MAPBL,RCON
125 EXTERN DBUF4,DBUF5,DBUF6,MBUF4,MBUF5,MBUF6,WAITX,SCOUNT
126 EXTERN GLNN,PALLN,CONID,DOCON,SHCST,SHPLL,DODIS,DISC,SEVNT
127 EXTERN EWAIT,CWAIT,MPBUF,UMPBUF,SCSERR,RBSIZ,MINMS,MINDS
128 EXTERN LNODE,LNODEN,GPRS,GPRSB
129
130 ;Externs found in DFCIB2
131
132 EXTERN ROFSET,LOFSET,OTHNOD,PKTMLT,PKTSIZ,MOVTYP,EXTEND
133 EXTERN SNDMSG,BLDCTP,GENLEN,OPCODE,READM,STADR,IMGDAT,BUFLEN
134 EXTERN REQUEM,EXAMNM,RETBUF,BUFTYP,GENFUN,GNCNST,DELAY,RCOUNT
135 EXTERN CTPPKT,SCSCMD,SCSRSP,SCSEVT,SAVREQ,SAVCTP,SAVSCS,SCSSAV
136 EXTERN ACTFLG,BUFLNM,BUFRNM,SAVRSP,READD,SNDDAT,REQUED,BFLNM1
137
138 ;Externs found in DFCIB3
139
140 EXTERN LISTEN,ACCEPT,RESPR,RDATR,RDATE,DATSE,DATSR
141
142 ;Externs found in DFCIBY
143
144 EXTERN TST99
145
146 ;Externs found in DFCIBM
147
148 EXTERN TST01,TST02
149 EXTERN TST10,TST11,TST12,TST13,TST14
150 EXTERN TST30,TST31,TST32,TST33,TST40
151 EXTERN TST50,TST51,TST52,TST53,TST54,TST55,TST56,TST57,TST58,TST59
152 EXTERN TST80
153 EXTERN TALL,BALL,MALL,DALL,BUFALL,CNTALL,XRCSE,DEF
```



```

154          SUBTTL  GLXLIB Interface
155
156      ;IB - the GLXLIB Interface Block
157
158      000000'  IB:      $BUILD  IB.SZ          ;Here to build the init block
159                  $SET      (IB.PRG,,%.MOD)    ;Program name
160                  $SET      (IB.OUT,,BOUT%%)   ;Default output routine
161                  $SET      (IB.FLG,IT.OCT,1)   ;Set open command terminal flag
162                  $EOB                      ;End of block
163
164      ;CSB template, BLT'd into place at top level parsing
165
166      000006'  CSBTMP: $BUILD  .CMGJB+1
167                  $SET      .CMFLG,RHMASK,REPARS ;Repars address
168                  $SET      .CMBFP,<POINT 7,BUFFER> ;Pointer to input buffer
169                  $SET      .CMPTR,<POINT 7,BUFFER> ;Start the parse at the beginning
170                  $SET      .CMCNT,,BUFSIZ*5-1 ;Size of buffer
171                  $SET      .CMABP,<POINT 7,ATMBUF> ;Pointer to start of atom buffer
172                  $SET      .CMABC,,ATMSZ*5-1 ;Size of atom buffer
173                  $SET      .CMRTY,<Point 7,PROMPT> ;Prompt (retry) pointer
174                  $SET      .CMGJB,,GJFBLK      ;GTJFN block
175                  $EOB
176
177      ;Program prompt
178
179      000020' 104 106 103 111 102 0  PROMPT: BYTE(7)'D','F','C','I','B',76,0 ;DFCIBalglebracket
180
181      ;Fobs and FBs for opening the take file
182
183      000022'  TAKFOB: $BUILD  (FOB.SZ)          ;File open block
184                  $SET      (FOB.FD,,TAKFD)     ;File descriptor area
185                  $SET      (FOB.CW,FB.BSZ,7)    ;Seven bit bytes
186                  $EOB
187
188      000027'  TAKFD:  $BUILD  (FDXSIZ)          ;FD
189                  $SET      (.FDLEN,FD.LEN,FDXSIZ) ;Set size of block
190                  $EOB
191

```



```

192
193 ;FOBs and FDs for opening the LOG file.
194
195 000123' LOGFOB: $BUILD (FOB.SZ) ;File open block
196 $SET (FOB.FD,,LOGFD) ;File descriptor area
197 $SET (FOB.CW,FB.BSZ,7) ;Seven bit bytes
198 $EOB
199
200 000130' LOGFD: $BUILD (FDXSIZ) ;FD size
201 $SET (.FDLEN,FD.LEN,FDXSIZ) ;Set size of block
202 $EOB
203
204 ;FOBs and FBs for opening the help file
205
206 000224' HFOB20: $BUILD (FOB.SZ) ;File open block
207 $SET (FOB.FD,,HLPFD2) ;File descriptor area
208 $SET (FOB.CW,FB.BSZ,7) ;Seven bit bytes
209 $EOB
210
211 000231' HLPFD2: $BUILD (.FDSTG+3) ;FD for TOPS-20
212 $SET (.FDLEN,FD.LEN,.FDSTG+3) ;Set size of block
213 $SET (.FDSTG,<ascii/dsk:d/>) ;First 5 characters of filename
214 $SET (.FDSTG+1,<ascii/fcib./>) ;Second 5
215 $SET (.FDSTG+2,<ascii/l:lp/>) ;Last 3
216 $EOB
217
218 000235' HFOB10: $BUILD (FOB.SZ) ;File open block
219 $SET (FOB.FD,,HLPFD1) ;File descriptor area
220 $SET (FOB.CW,FB.BSZ,7) ;Seven bit bytes
221 $EOB
222
223 000242' HLPFD1: $BUILD (.FDPPN+1) ;FD for TOPS-10
224 $SET (.FDLEN,FD.LEN,.FDPPN+1) ;Set size of block
225 $SET (.FDSTR,<sixbit/DSK/>) ;Structure
226 $SET (.FDNAM,<sixbit/DFCIB/>) ;Filename
227 $SET (.FDEXT,<sixbit/HLP/>) ;Extension
228 $EOB
229
230 ;TOPS-10 Interrupt Vector
231
232 PIVECT: CCOFFS=.-PIVECT ;Offset : control C trap
233 000247' 000000 003206' EXP CCIN10 ;.PSVNP New PC
234 000250' 000000 000000 EXP 0 ;.PSVOP Old PC
235 000251' 000000 777775 XWD 0,.PCSTP ;.PSVFL ^C interrupts
236 000252' 000000 000000 EXP 0 ;.PSVIS interrupt status

```

```

237          SUBTTL General Storage
238
239          ;Program flags and counters, these flags and counters are used internally
240
241          000253'      PDL:      BLOCK      <PLEN==100>          ;The stack
242
243          000353'      TSTNAM: BLOCK      1          ;Pointer to current ASCIZ test name
244          000354'      TSTDSC: BLOCK      1          ;Pointer to current ASCIZ test descr
245          000355'      PASCNT: BLOCK      1          ;Pass counter counting up
246          000356'      PASSCN: BLOCK      1          ;Number of passes to run this RUN cmd
247          000357'      ITCNT: BLOCK      1          ;Iterations count
248          000360'      TITCNT: BLOCK      1          ;Number of iterations to run this test
249
250          000361'      PATHS:  BLOCK      1          ;Default to auto path select
251          000362'      NODEN:  BLOCK      1          ;Node number stored here
252          000363'      CHKEV:  BLOCK      1          ;Event to check stored here
253
254          000364'      MAPBBA: BLOCK      1          ;MAPBB address (pages from GLXMEM)
255          000365'      MONTYP: BLOCK      1          ;Monitor type, -1 if TOPS-20
256          000366'      OPSNUM: BLOCK      1          ;20 if a 20, 10 if a 10
257          000367'      CPUTYP: BLOCK      1          ;CPU type in sixbit
258          000370'      CPUSER: BLOCK      1          ;CPU serial number
259          000371'      LOGIFN: BLOCK      1          ;IFN of log file, 0 if not logging
260          000372'      TAKIFN: BLOCK      1          ;IFN to take file, 0 if not taking now
261
262          000373'      LCKFLG: BLOCK      1          ;-1=Locked in core (TOPS-10)
263          000374'      SWTRAC: BLOCK      1          ;Trace switch set if -1
264          000375'      SWPALL: BLOCK      1          ;Print all errors set if -1
265          000376'      SWHALT: BLOCK      1          ;Halt on error set if -1
266          000377'      SWBELL: BLOCK      1          ;Dink on error if -1
267
268          ;Parser storage
269
270          000400'      CSB:      BLOCK      .CMGJB+1          ;The real CSB is here
271          000412'      BUFFER:  BLOCK      <BUFSIZ==500>      ;Command buffer
272          001112'      ATMBUF:  BLOCK      <ATMSZ==100>      ;Atom buffer
273          001212'      GJFBLK:  BLOCK      <GJFSIZ==.GJRTY+2> ;GTJFN block
274
275          ;Help file reader storage
276
277          001230'      HLPBUF:  BLOCK      ATMSZ          ;Atom buffer for help request
278          001330'      HLPLIN:  BLOCK      ^D135/5        ;Line of help text copied here
279          001363'      HLPEOF:  BLOCK      1              ;-1=EOF on help file
280          001364'      HLPIFN:  BLOCK      1              ;IFN of help file
281
282          ;This is the run table, for remembering the tests to run
283          ;repeat for each test: first word: address_of_test_start,,iteration count
284          ;                                second word: 0,,address_of_ASCIZ_test_name
285          ;zero word terminates run table
286
287          001365'      MAINBF:  BLOCK      1+<MAINBL==^D40> ;Buffer for dispatch table.
288          001436'      SNODET:  BLOCK      ^D16            ;-1=node selected, 0=not selected
  
```

SUBTTL Parse Tables

;Here are all the tables to be parsed

;This is the top level command table.

289					
290					
291					
292					
293					
294					
295	001456'	000032	000032	MAINCM: \$STAB	;Start command table, top level
296	001457'	004622'	004176'	KEYTAB	.ACCEP,ACCEPT
297	001460'	004624'	003365'	KEYTAB	.BUILD,BUILD
298	001461'	004626'	002037'	C..CLE: KEYTAB	.CLEAR,CLEAR
299	001462'	004630'	003251'	KEYTAB	.CONN,CONNECT
300	001463'	004632'	004202'	KEYTAB	.DDT,DDT
301	001464'	004633'	002257'	KEYTAB	.DESEL,DESELECT
302	001465'	004635'	003254'	KEYTAB	.DCONN,DISCONNECT
303	001466'	004640'	003212'	KEYTAB	.QUIT,EXIT,CM%INV
304	001467'	004642'	002310'	KEYTAB	.HELP,HELP
305	001470'	004643'	004172'	KEYTAB	.LISTE,LISTEN
306	001471'	004645'	004133'	KEYTAB	.MAPB,MAP-BUFFER
307	001472'	004650'	004156'	KEYTAB	.PAUSE,PAUSE
308	001473'	004652'	003212'	KEYTAB	.QUIT,QUIT
309	001474'	004653'	004013'	KEYTAB	.READ,READ
310	001475'	004654'	004245'	KEYTAB	.DREQ,REQUEST-DATA
311	001476'	004657'	004033'	KEYTAB	.REQ,REQUEUE
312	001477'	004661'	004047'	KEYTAB	.RRET,RETURN-MESSAGE
313	001500'	004664'	002477'	KEYTAB	.RUN,RUN
314	001501'	004665'	002213'	KEYTAB	.SELEC,SELECT
315	001502'	004667'	003217'	KEYTAB	.SEND,SEND
316	001503'	004670'	002061'	C..SET: KEYTAB	.SET,SET
317	001504'	004671'	002663'	C..SHO: KEYTAB	.SHOW,SHOW
318	001505'	004672'	003073'	KEYTAB	.TAKE,TAKE
319	001506'	004673'	004052'	KEYTAB	.TYPE,TYPE
320	001507'	004674'	004152'	KEYTAB	.UMAPB,UNMAP-BUFFER
321	001510'	004677'	003257'	KEYTAB	.WAIT,WAIT
322			000033	\$ETAB	;End command table, top level

323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353

001511' 000005 000005
 001512' 004700' 002141'
 001513' 004702' 002127'
 001514' 004705' 002065'
 001515' 004707' 002113'
 001516' 004711' 002100'
 000006
 000020 000020
 001520' 004715' 002671'
 001521' 004722' 003006'
 001522' 004726' 003032'
 001523' 004730' 003070'
 001524' 004733' 003011'
 001525' 004740' 002732'
 001526' 004744' 003015'
 001527' 004750' 003025'
 001530' 004702' 002701'
 001531' 004755' 003040'
 001532' 004761' 003065'
 001533' 004767' 002767'
 001534' 004771' 002735'
 001535' 004762' 003035'
 001536' 004774' 002712'
 001537' 004707' 002754'
 000021

;Here is the table that for the SET command

SETTAB: \$STAB ;Start of the set table
 KEYTAB .SLOG,LOGGING ;Set logging <filename>
 KEYTAB .SL3,PATH-SELECTION ;Set path selection
 KEYTAB .SWITS,SWITCH ;Switches
 KEYTAB .SL2,TIME-OUT ;Set length of time out
 KEYTAB .SL1,TYPE-OUT-LINE-LIMIT ;Set typeout line limit length
 \$ETAB ;End of set table

;This table is used with the 'SHOW' command.

SHOTAB: \$STAB ;Start of SHOW command table
 KEYTAB SHALL,ALL-PROGRAM-PARAMETERS ;Show state of program parameters
 KEYTAB SHCIC,CI-CONFIGURATION ;Show CI Configuration
 KEYTAB SHCO,COUNTERS ;Show counters
 KEYTAB SHEVNT,EVENT-QUEUE ;Show event queue
 KEYTAB SHCID,ID-OF-CONNECTED-NODE ;Show connect id
 KEYTAB SLNODE,LOCAL-NODE-NUMBER ;Show local node number
 KEYTAB SHMBN,MAPPED-BUFFER-NAME ;Routine to show mapped buffer name
 KEYTAB SHMBS,MINIMUM-BUFFER-SIZES ;Show minimum buffer sizes
 KEYTAB SHSPS,PATH-SELECTION ;Show path selection
 KEYTAB SHPS,PATH-STATUS-OF-NODE ;Show path status of a node
 KEYTAB SHPST,POLL-STATUS-OF-CONNECTED-NODE ;Show poll status
 KEYTAB SHRTBL,RUN-TABLE ;Show entries in run table
 KEYTAB SHSN,SELECTED-NODES ;Routine to show selected nodes
 KEYTAB SHSOCN,STATUS-OF-CONNECTED-NODE ;Show connect status
 KEYTAB SHSWI,SWITCHES ;Show state of switches
 KEYTAB SHTIOU,TIME-OUT ;Show length of time out
 \$ETAB ;End of SHOW command table


```

354
355 ;Confirm block, should be the only one in the program
356
357 001540' 010000 000000 CFM: FLDDB.(.CMCFM)
358 001541' 000000 000000
359
360 ;This table is used with the 'CLEAR' command.
361
362 001542' 000002 000002 CLRTAB: $STAB ;Start of clear command table
363 001543' 004700' 002056' KEYTAB .CLOG,LOGGING ;Clear logging
364 001544' 004705' 002043' KEYTAB .SWITC,SWITCH ;Clear switches
365 000003 SETAB ;End of clear command table
366
367 ;This table is used with the CLEAR SWITCH and SET SWITCH commands.
368
369 001545' 000004 000004 SWITAB: $STAB ;Begining of switch name table
370 001546' 004776' 000377' KEYTAB SWBELL,BELL ;Dink on error
371 001547' 004777' 000376' KEYTAB SWHALT,HALT ;Halt on an error
372 001550' 005000' 000375' KEYTAB SWPALL,PALL ;Print all errors
373 001551' 005001' 000374' KEYTAB SWTRAC,TRACE ;Report each test before execution
374 000004 SWITCHN==.-SWITAB-1 ;Number of switches in table
375 000005 SETAB ;End of switch name table
376
377 ;Tables for SELECT NODE command
378
379 001552' 001004 001555' SETNN: FLDDB.(.CMNUM,,^D10,<node number, 0-15>,,SNALL)
380 001553' 000000 000012
381 001554' 44 07 0 00 005003'
382 001555' 000000 000000 SNALL: FLDDB.(.CMKEY,,SNTAB)
383 001556' 000000 001557'
384 001557' 000001 000001 SNTAB: $STAB ;Start of the select node ALL table
385 001560' 005007' 002243' KEYTAB .SELNA,ALL ;Routine to select node ALL
386 000002 SETAB ;End of select table
  
```



```

387
388
389
390 001561' 000030 000030 TSTTAB: $STAB ;Start of test table
391 001562' 005010' 000000* KEYTAB TST01,TST01 ;Basic tests
392 001563' 005012' 000000* KEYTAB TST02,TST02
393
394 001564' 005014' 000000* KEYTAB TST10,TST10 ;Message tests
395 001565' 005016' 000000* KEYTAB TST11,TST11
396 001566' 005020' 000000* KEYTAB TST12,TST12
397 001567' 005022' 000000* KEYTAB TST13,TST13
398 001570' 005024' 000000* KEYTAB TST14,TST14
399
400 001571' 005026' 000000* KEYTAB TST30,TST30 ;Data tests
401 001572' 005030' 000000* KEYTAB TST31,TST31
402 001573' 005032' 000000* KEYTAB TST32,TST32
403 001574' 005034' 000000* KEYTAB TST33,TST33
404 001575' 005036' 000000* KEYTAB TST40,TST40
405
406 001576' 005040' 000000* KEYTAB TST50,TST50 ;Buffer tests
407 001577' 005042' 000000* KEYTAB TST51,TST51
408 001600' 005044' 000000* KEYTAB TST52,TST52
409 001601' 005046' 000000* KEYTAB TST53,TST53
410 001602' 005050' 000000* KEYTAB TST54,TST54
411 001603' 005052' 000000* KEYTAB TST55,TST55
412 001604' 005054' 000000* KEYTAB TST56,TST56
413 001605' 005056' 000000* KEYTAB TST57,TST57
414 001606' 005060' 000000* KEYTAB TST58,TST58
415 001607' 005062' 000000* KEYTAB TST59,TST59
416
417 001610' 005064' 000000* KEYTAB TST80,TST80 ;Counter test
418
419 001611' 005066' 000000* KEYTAB TST99,TST99 ;Exerciser
420
421 000031 $ETAB ;End of test table
422
423 ;This table is the SCRIPT table and used with the 'RUN' command.
424
425 001612' 000010 000010 SCRTAB: $STAB ;Start of script table
426 001613' 005070' 000000* KEYTAB TALL,ALL-TESTS
427 001614' 005072' 000000* KEYTAB BALL,BASIC-TESTS
428 001615' 005075' 000000* KEYTAB BUFALL,BUFFER-TESTS
429 001616' 005100' 000000* KEYTAB CNTALL,COUNTER-TEST
430 001617' 005103' 000000* KEYTAB DALL,DATA-TESTS
431 001620' 005106' 000000* KEYTAB DEF,DEFAULT ;Default script
432 001621' 005110' 000000* KEYTAB XRCER,EXERCISER
433 001622' 005112' 000000* KEYTAB MALL,MESSAGE-TESTS
434 000011 $ETAB ;End of script table
  
```

```

435
436 ;This area contains the macros to build the command descriptor block
437
438 ;Tables for RUN command
439
440 001623' 000004 001626' RWPARS: FLDDDB.(.CMKEY,,SCRTAB,<a test script,>,,TSPAR2)
441 001624' 000000 001612'
442 001625' 44 07 0 00 005115'
443 001626' 000004 000000 TSPAR2: FLDDDB.(.CMKEY,,TSTTAB,<a specific test,>)
444 001627' 000000 001561'
445 001630' 44 07 0 00 005120'
446
447 001631' 003000 001633' RPPARS: FLDDDB.(.CMSWI,,PASTAB,,,RUPARS)
448 001632' 000000 001650'
449 001633' 000004 001636' RUPARS: FLDDDB.(.CMKEY,,SCRTAB,<a test script,>,,TSPAR1)
450 001634' 000000 001612'
451 001635' 44 07 0 00 005115'
452 001636' 000004 001540' TSPAR1: FLDDDB.(.CMKEY,,TSTTAB,<a specific test,>,,CFM)
453 001637' 000000 001561'
454 001640' 44 07 0 00 005120'
455
456 001641' 013004 001644' RVPARS: FLDDDB.(.CMCMA,,,<Comma followed by another test name>,,ISPARS)
457 001642' 000000 000000
458 001643' 44 07 0 00 005124'
459 001644' 003000 001540' ISPARS: FLDDDB.(.CMSWI,,ITSTAB,,,CFM)
460 001645' 000000 001646'
461
462 001646' 000001 000001 ITSTAB: $STAB ;Start of the switch table
463 001647' 005134' 000000 KEYTAB 0,ITERATIONS: ;Iterations count switch
464 000002 $ETAB ;End of switch table
465
466 001650' 000001 000001 PASTAB: $STAB ;Start of pass count table
467 001651' 005137' 000000 KEYTAB 0,PASSES: ;Pass count
468 000002 $ETAB ;End of pass count table

```

```

469
470 ;Help command parse blocks
471
472 ;HELP command
473
474 001652'
475 001652' 000004 001655'
476 001653' 000000 001456'
477 001654' 44 07 0 00 005141'
478 001655' 000004 001660'
479 001656' 000000 001612'
480 001657' 44 07 0 00 005146'
481 001660' 000004 001663'
482 001661' 000000 001561'
483 001662' 44 07 0 00 005153'
484 001663' 023005 001540'
485 001664' 44 07 0 00 005157'
486 001665' 44 07 0 00 005160'
487
488
489
490 001666' 001461' 005166'
491 001667' 001503' 005170'
492 001670' 001504' 005172'
493 000003'
494 001671' 000000 001540'

;Table for additional help parsing.

HLPARS:
HCS: FLDDB.(.CMKEY,,MAINCM,<Help about a command,>,,HSC)

HSC: FLDDB.(.CMKEY,,SCRTAB,<Help about a script,>,,HTL)

HTL: FLDDB.(.CMKEY,,TSTTAB,<Help about a test,>,,HAL)

HAL: FLDDB.(.CMTOK,CM%SDH,<POINT 7,[ASCIZ/*/]>,<'*' for all of the help file>,,CFM)

;Table for additional help parsing.

HLPADD: XWD C..CLE,[FLDDB.(.CMKEY,,CLRTAB,,,CFM)] ;CLEAR
        XWD C..SET,[FLDDB.(.CMKEY,,SETTAB,,,CFM)] ;SET
        XWD C..SHO,[FLDDB.(.CMKEY,,SHOTAB,,,CFM)] ;SHOW
HLPADL==.-HLPADD
XWD 000000,CFM ;None of the above

```

495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537

```

: *****
: *      TOP level command parser      *
: *****

```

```

: The macro used to build the command descriptor block is (FLDDB.) and has the
: following format (type, flags, data, help, default, additional command data block),
: the type field will contain the COMMAND FUNCTION CODE, the function codes are:

```

.CMKEY= 0	: Keyword	.CMUSR= 12	: User name
.CMNUM= 1	: Number	.CMCMA= 13	: Comma
.CMNOI= 2	: Guide (noise) word	.CMINI= 14	: Init line
.CMSWI= 3	: Switch	.CMDEV= 16	: Device name
.CMIFI= 4	: Input file	.CMTXT= 17	: Text to action char
.CMOFI= 5	: Output file	.CMQST= 21	: Quoted string
.CMFIL= 6	: General filespec	.CMUQS= 22	: Unquoted string
.CMFLD= 7	: Arbitrary field	.CMTOK= 23	: Token
.CMCFM= 10	: Confirm	.CMNUX= 24	: Number delimited
.CMDIR= 11	: Directory name		: by non-digit

```

: The flag field will contain one of the following flags or will be represented
: by two commas (,,), this indicates to the macro that this field is blank.

```

```

: The flags are:

```

CMZHPP	: Help pointer is present
CMZDPP	: Default pointer is present
CMZSDH	: Suppress default help message

```

: The data field is dependent on the command function, see TOPS20 monitor call
: manual COMMD JSYS v544. This field may be omitted by using two commas (,,).

```

```

: The help field points to a message to be printed if the question mark (?)
: is typed. This field may be omitted by using two commas (,,).

```

```

: The default field is pointer to a string to be used if the ESCAPE is the first
: character to be typed. This field may be omitted by using two commas (,,).

```

```

: The alternate command block is a pointer to a command block to be parsed if
: the parse failed in the first command block. This field may be omitted.

```

```

: The function field is always used but not all other fields are necessarily
: used. If a field to the right is to be used, the unused fields separating
: the fields must be represented by two commas (,,).

```



```

538          SUBTTL Initial Start Up
539
540          ;Determine if TOPS-10 or TOPS-20, and get GLXLIB going
541
542          SETUP: MOVE P,[IOWD PLEN,PDL]      ;Load stack pointer
543                  $CALL UNLKIT              ;Incase ^C, START done
544                  MOVE S1,[112,,11]         ;Load %CNMNT monitor type word
545                  GETTAB S1,                ;This will not call PA1050 on the -20
546                  SETZ S1,                 ;Oops
547                  TXNE S1,4823              ;Skip if TOPS-10
548                  SKIPA S1,[XWD ,PRIIN,,PRIOU] ;TOPS-20, set up input/output JFNs
549                  SKIPA S1,[XWD 377776,377777] ;TOPS-10, set up fake i/o JFNs
550                  SKIPA S2,[DEC 20]         ;TOPS-20, load a twenty
551                  SKIPA S2,[DEC 10]         ;TOPS-10, load a ten
552                  SETOM MONTYP              ;TOPS-20, set the flag indicating that
553                  MOVEM S1,CSBTMP+.CMIOJ    ;Set I/O JFNs properly for -10 or -20
554                  MOVEM S2,OPSNUM           ;Set ten or twenty
555
556          ;Get GLXLIB up.
557
558          MOVEI S1,IB.SZ                    ;Load size of the init block
559          MOVEI S2,IB                       ; and point to it
560          $CALL I$INIT                      ;Init the GLXLIB stuff
561
562          ;Get transmit buffer if not already known, release log file IFN if restarted
563
564          SKIPE MAPBBA                      ;Skip if MAPBB already there
565          JRST SETUP3                      ;Yes
566          MOVEI S1,4                        ;Get four pages
567          $CALL M$AQNP                      ;Get a page from the GLXLIB allocator
568          PG2ADR S1                          ;Get the address from it
569          MOVEM S1,MAPBBA                   ;Save it as transmit buffer
570
571          ;Close log file if needed upon restart
572
573          SETUP3: SKIPE S1,LOGIFN            ;Is there a log file open?
574                  $CALL F$REL               ;Release the IFN
575                  SETZM LOGIFN              ;There sure isn't a log file now
576
577          ;Announce ourselves, set up data areas
578
579          $CALL CPUNUM                      ;Get CPU number, etc.
580          $TEXT (<
581          ^I/ANNOUN/^H/[-1]/
582          >)
583
584          ;Announce ourselves

```



```

584
585                                     ;Enable capabilities if TOPS-20
586
587 001726' 336 00 0 00 000365'      SKIPN  MONTYP      ;TOPS-20?
588 001727' 254 00 0 00 001741'      JRST   SETUP1      ;Nope
589
590 001730' 201 01 0 00 400000      MOVEI   S1,..FHSLF      ;Set up for current process
591 001731' 104 00 0 00 000150      RPCAP%      ;Get my capabilities
592 001732' 606 02 0 00 640000      TRNN     S2,SC%WHL!SC%OPR!SC%MNT ;Any good capabilities present?
593 001733' 260 17 0 00 001724*     $TEXT   (,<% Not enough capabilities to run diagnostic - proceeding>)
594 001735' 201 01 0 00 400000      MOVEI   S1,..FHSLF      ;Incase called above
595 001736' 200 03 0 00 000002      MOVE     T1,S2      ;Enable all capabilities
596 001737' 104 00 0 00 000151      EPCAP%      ;Enable TOPS-20 capabilities
597 001740' 254 00 0 00 002020'      JRST    TOPLVL      ;Enter top level parser
598
599                                     ;Here if TOPS-10 to set up interrupt system
600
601 001741' 201 01 0 00 000247'      SETUP1: MOVEI   S1,PIVECT      ;Point to PI vector
602 001742' 047 01 0 00 000135      PIINI.   S1,          ;Get it to the monitor
603 001743' 260 17 0 00 001733*     $TEXT   (,<? Can't use PI system - error code ^0/S1/>)
604 001745' 200 01 0 00 005256'      MOVE     S1,[PS.FAC![EXP .PCSTP,<CCOFFS,,0>,0]] ;Control-C interrupts
605 001746' 047 01 0 00 000136      PISYS.   S1,          ;or CALLI AC,136
606 001747' 260 17 0 00 001743*     $TEXT   (,<? Can't activate control-C interrupts - error code ^0/S1/>)
607 001751' 205 01 0 00 100000      MOVX     S1,PS.FON      ;Turn on interrupt system
608 001752' 047 01 0 00 000136      PISYS.   S1,          ;or CALLI AC,136
609 001753' 260 17 0 00 001747*     $TEXT   (,<? Can't turn on PI - error code ^0/S1/>)
610 001755' 254 00 0 00 002020'      JRST    TOPLVL      ;Start parsing

```

```

611
612 ;Here to get the CPU number and so on from the monitor.
613
614 001756' 260 17 0 00 000000* CPUNUM: $CALL .CPUTY ;Ask GLXLIB for CPU type
615 001757' 301 01 0 00 000002 CAIL S1,2 ;Is it
616 001760' 303 01 0 00 000005 CAILE S1,5 ; in known range?
617 001761' 634 02 0 00 000002 TDZA S2,S2 ;Nope, say nothing
618 001762' 200 02 0 01 005313' MOVE S2,[EXP 'KI10 ','KL10' ;'KS10 'J-2(S1)
619 001763' 202 02 0 00 000367' MOVEM S2,CPUTYP ;Save sixbit CPU type
620
621 001764' 332 00 0 00 000365' SKIPE MONTYP ;Skip if TOPS-10
622 001765' 254 00 0 00 001773' JRST CPUN20 ;Tops-20
623
624 001766' 200 01 0 00 005320' MOVE S1,[20,,11] ;Load %CNSER CPU0 serial number
625 001767' 047 01 0 00 000041 GETTAB S1, ;Get it
626 001770' 400 01 0 00 000000 SETZ S1, ;Error- punt it off
627 001771' 202 01 0 00 000370' MOVEM S1,CPUSER ;Save it there
628 001772' 263 17 0 00 000000 $RET ;Return to caller
629
630 001773' 201 01 0 00 000034 CPUN20: MOVEI S1,APRID ;Load table for APRID number
631 001774' 104 00 0 00 000010 GETAB% ;Get it
632 001775' 400 01 0 00 000000 SETZ S1, ;Error? punt
633 001776' 202 01 0 00 000370' MOVEM S1,CPUSER ;Save it
634 001777' 263 17 0 00 000000 $RET ;Return
635
636 ;Here is the text string output to terminal and log file when we are started.
637
638 ANNOUN: ITEXT (<DFCIB Computer Interconnect/Cluster Test Protocol Exerciser
639 Version ^V/.JBVER##/, ^W/CPUTYP/, TOPS-^D/OPSNUM/, CPU#-^D/CPUSER/
640 >)
641

```

```

642          SUBTTL Top Level Parser
643
644          ;Here on a reparse.
645
646          002016' 202 17 0 00 005174'  REPARS: MOVE    P,[IOWD PLEN,PDL]      ;Here on reparse - reset stack
647          002017' 254 00 0 00 002032'  JRST      PRSCM3      ; and restart parsing
648
649          ;Here for top level commands, reloads the stack and parses a new command.
650
651          002020' 200 17 0 00 005174'  TOPLVL: MOVE    P,[IOWD PLEN,PDL]      ;Point to the stack (again)
652          002021' 200 01 0 00 005345'  MOVE      S1,[XWD CSBTMP,CSB]  ;Initialize the CSB by BLTing
653          002022' 251 01 0 00 000411'  BLT       S1,CSB+.CMGJB  ; the CSB template over it
654
655          ;Normal commands loop.
656
657          002023' 336 01 0 00 000372'  PRSCM1: SKIPN   S1,TAKIFN      ;TAKE command in progress?
658          002024' 254 00 0 00 002030'  JRST      PRSCM2      ;Not in a take command
659          002025' 506 01 0 00 000401'  HRLM      S1,CSB+.CMIOJ  ;Take command, set the input JFN
660          002026' 200 01 0 00 005346'  MOVE      S1,[Point 7,[0]] ;Point to a zero prompt
661          002027' 202 01 0 00 000402'  MOVEM     S1,CSB+.CMRTY  ;Reset the prompt to be nothing
662
663          002030' 201 02 0 00 005347'  PRSCM2: MOVEI   S2,[FLDDB. (.CMINI)] ;Load address of init block
664          002031' 260 17 0 00 004443'  $CALL     CMDPRS      ;Init parser
665
666          ;Enter here on reparse (after the CMINI call). Parse a keyword and dispatch.
667
668          002032' 201 02 0 00 005350'  PRSCM3: MOVEI   S2,[FLDDB. (.CMKEY,MAINCM)] ;Point to top level parsing block
669          002033' 260 17 0 00 004443'  $CALL     CMDPRS      ;parse top level command
670          002034' 550 01 0 02 000000'  HRRZ      S1,(S2)      ;Get the keyword address
671          002035' 260 17 0 01 000000'  $CALL     (S1)      ;Dispatch
672          002036' 254 00 0 00 002023'  JRST      PRSCM1      ;Get another command
  
```

```

673          SUBTTL  Clear Command
674
675          ;Tables and parsing for the clear command.
676          ;The clearing of the switches is done in this parser, no external routine
677          ;required.
678
679          002037' 201 02 0 00 005352' .CLEAR: MOVEI S2,[FLDDB.(.CMKEY,,CLRTAB)] ;Clear parse table
680          002040' 260 17 0 00 004443'          $CALL CMDPRS ;Parse it
681          002041' 550 01 0 02 000000          HRRZ S1,(S2) ;Get the keyword address
682          002042' 324 17 0 01 000000          PJRST (S1) ;Go to clear routine
683
684          ;Here for CLEAR SWITCH command
685
686          002043' 201 02 0 00 005354' .SWITC: MOVEI S2,[FLDDB.(.CMKEY,,SWITAB)] ;Switch table
687          002044' 260 17 0 00 004443'          $CALL CMDPRS ;Parse the switch
688          002045' 550 01 0 02 000000          HRRZ S1,(S2) ;Get the address with this keyword
689          002046' 261 17 0 00 000001          PUSH P,S1 ;Save S1 and continue parse
690          002047' 201 02 0 00 005365'          MOVEI S2,[FLDDB.(.CMCMA,,, <Comma to clear additional switches>,,CFM)]
691          002050' 260 17 0 00 004443'          $CALL CMDPRS ;Parse it
692          002051' 302 03 0 00 001540'          CAIE T1,CFM ;Was last character a comma ?
693          002052' 260 17 0 00 002043'          $CALL .SWITC ;Loop for more of them
694          002053' 262 17 0 00 000001          POP P,S1 ;Get address to reset
695          002054' 402 00 0 01 000000          SETZM (S1) ;Clear it
696          002055' 263 17 0 00 000000          $RET ;Return
697
698          ;Here for CLEAR LOGGING
699
700          002056' 336 00 0 00 000371' .CLOG: SKIPN LOGIFN ;Skip if a log file there
701          002057' 254 00 0 00 005402'          CMDERR <Logging was not enabled>
702          002060' 324 17 0 00 004402'          PJRST LOGCLS ;Close the log file
  
```

```

703          SUBTTL Set Command
704
705          ;Parsing for the SET command.
706
707          .SET: MOVEI S2,[FLDDB.(.CMKEY,,SETTAB)] ;SET parse table
708                SCALL CMDPRS ;Parse it
709                HRRZ S1,(S2) ;Get the keyword address
710                PJRST (S1) ;Go to clear routine
711
712          ;Here for SET SWITCH
713
714          .SWITS: MOVEI S2,[FLDDB.(.CMKEY,,SWITAB)] ;Set Switch parse table
715                  SCALL CMDPRS ;Parse it
716                  HRRZ S1,(S2) ;Get the bits for this keyword
717                  PUSH P,S1 ;Save switch on stack
718                  MOVEI S2,[FLDDB.(.CMCMA,,,<comma to set additional switches>,,CFM)]
719                  SCALL CMDPRS ;Parse it
720                  CAIE T1,CFM ;Was last character a comma ?
721                  SCALL .SWITS ;Yes, get another switch
722                  POP P,S1 ;Get the address back
723                  SETOM (S1) ;Set the switch
724                  SRET ;Return
725
726          ;SET TYPE-OUT-LINE-LIMIT
727
728          .SL1: GUIDE <number of lines to display buffer contents> ;Noise word
729
730                  MOVEI S2,[FLDDB.(.CMNUM,,^D10,<Number of lines of output, 1-40>)]
731                  SCALL CMDPRS ;Get the number into S2
732                  MOVE S1,[^D1,,^D40] ;No, is this number in range, 1-40 ?
733                  SCALL RANGCK ;Range check the number
734                  CMDERR <Number of lines out of range> ;Out of range
735                  SCALL CMDCFM ;We expect command confirmation here
736                  MOVEM S2,TYLEN ;Save it
737                  SRET ;Return to toplevel
738
739          002112' 000000 000012 TYLEN: ^D10 ;Default to 10 lines
  
```



```

740
741 ;SET TIME-OUT
742
743 002113' 201 02 0 00 005472' .SL2: GUIDE <in seconds> ;Noise word
744 002114' 260 17 0 00 004443'
745 002115' 201 02 0 00 005502' MOVEI S2,[FLDDB.(.CMNUM,,^D10,<time out in seconds, 1-2000>)]
746 002116' 260 17 0 00 004443' $CALL CMDPRS ;Get the number into S2
747 002117' 200 01 0 00 005505' MOVE S1,[^D1,,^D2000] ;No, is this number in range, 1-2000 ?
748 002120' 260 17 0 00 004417' $CALL RANGCK ;Range check the number
749 002121' 254 00 0 00 005520' CMDERR <Time out value out of range> ;out of range
750 002122' 260 17 0 00 004435' $CALL CMDCFM ;We expect command confirmation here
751 002123' 221 02 0 00 001750' IMULI S2,^D1000 ;Convert milliseconds to seconds
752 002124' 202 02 0 00 002126' MOVEM S2,SECS ;Save it
753 002125' 254 00 0 00 002755' JRST .STIOU ;Show it and return
754
755 002126' 000000 011610 SECS: ^D5000 ;Milliseconds to wait
756
757 ;Here for "SET PATH-SELECTION" command
758
759 002127' 201 02 0 00 005524' .SL3: MOVEI S2,[FLDDB.(.CMKEY,.SL3B)] ;Table to use
760 002130' 260 17 0 00 004443' $CALL CMDPRS ;Parse that please
761 002131' 550 01 0 02 000000 HRRZ S1,(S2) ;Load resulting keyword value
762 002132' 260 17 0 00 004435' $CALL CMDCFM ;We expect command confirmation here
763 002133' 202 01 0 00 000361' MOVEM S1,PATHS ;Save path to use
764 002134' 263 17 0 00 000000 $RET ;Return to toplevel
765
766 ;Here is the table for the path select command
767
768 002135' 000003 000003 SL3B: $STAB ;Start of the wait table
769 002136' 005526' 000001 KEYTAB .SSPTA,A-PATH ;Select A path
770 002137' 005530' 000002 KEYTAB .SSPTB,B-PATH ;Select B path
771 002140' 005532' 000000 KEYTAB .SSAPS,PATH-AUTO-SELECT ;Select auto select
772 000004 SETAB ;End of wait table
  
```

773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824

002141' 332 00 0 00 000371'
 002142' 254 00 0 00 005555'
 002143' 201 02 0 00 005563'
 002144' 260 17 0 00 004443'
 002145' 402 00 0 00 001212'
 002146' 200 01 0 00 005565'
 002147' 251 01 0 00 001227'
 002150' 336 00 0 00 000365'
 002151' 254 00 0 00 002177'

 002152' 205 01 0 00 400000'
 002153' 202 01 0 00 001212'
 002154' 200 01 0 00 000401'
 002155' 202 01 0 00 001213'
 002156' 561 01 0 00 005566'
 002157' 202 01 0 00 001216'
 002160' 561 01 0 00 005570'
 002161' 202 01 0 00 001217'
 002162' 561 01 0 00 005571'
 002163' 202 01 0 00 001214'
 002164' 201 02 0 00 005574'
 002165' 260 17 0 00 004443'
 002166' 260 17 0 00 004435'
 002167' 561 01 0 00 000131'

 002170' 200 03 0 00 005600'
 002171' 104 00 0 00 000030'
 002172' 320 16 0 00 002173'
 002173' 200 01 0 00 000002'
 002174' 104 00 0 00 000023'
 002175' 320 16 0 00 002176'
 002176' 254 00 0 00 004316'

;SET LOGGING command

```
.SLOG:  SKIPE  LOGIFN      ;Logging enabled now?
        CMDERR <Logging is already enabled, type CLEAR LOGGING first>
        MOVEI  S2,[FLDDB. (.CMNOI,,<Point 7,[ASCIZ/to file/]>)]
        $CALL  CMDPRS      ;Do it
        SETZM  GJFBLK      ;Clear first word
        MOVE   S1,[GJFBLK,,GJFBLK+1] ;Set up to clear block
        BLT    S1,GJFBLK+GJFSIZ-1    ;Clear the block
        SKIPN  MONTYP      ;Skip if TOPS-20
        JRST   .SLOG5     ;TOPS-10
```

;TOPS-20 SET LOGGING command

```
        MOVX   S1,GJ%FOU      ;File is for output
        MOVEM  S1,GJFBLK+.GJGEN ; into flags word
        MOVE   S1,CSB+.CMIOJ   ;Load I/O JFNs
        MOVEM  S1,GJFBLK+.GJSRC ; into block
        HRROI  S1,[ASCIZ/DFCIB/] ;Point at default file name
        MOVEM  S1,GJFBLK+.GJNAM ;Save for GTJFN
        HRROI  S1,[ASCIZ/LOG/]  ;Default extension
        MOVEM  S1,GJFBLK+.GJEXT ;Save in GTJFN block
        HRROI  S1,[ASCIZ/DSK/]  ;Get the default structure
        MOVEM  S1,GJFBLK+.GJDEV ;Save the device
        MOVEI  S2,[FLDDB. (.CMFIL,,,,<DFCIB.LOG>)] ;Output file type
        $CALL  CMDPRS      ;Hello GLXLIB
        $CALL  CMDCFM      ;Confirm that
        HRROI  S1,LOGFD+.FDSTG ;Point to the FD
        MOVX   T1,FLD(.JSAOF,JS%DEV)!FLD(.JSAOF,JS%DIR)!FLD(.JSAOF,JS%NAM)!FLD(.JSAOF,JS%TY
P)!FLD(.JSAOF,JS%GEN)!JS%PAF
        JFNS%      ;JFN to string
        ERJMP    .+1      ;Error, ignore it
        MOVE   S1,S2      ;Reload the JFN
        RLJFN%      ;Release it please
        ERJMP    .+1      ;Error? Punt it
        JRST   LOGOPN     ;Open the logging file and return
```

;Here for TOPS-10 ENABLE LOGGING command

```
.SLOG5: MOVE   S1,[SIXBIT/DFCIB/] ;Get file name
        STORE  S1,GJFBLK+.FDNAM    ;Save in default block
        MOVSI  S1,'CMD'           ;Get default extension
        STORE  S1,GJFBLK+.FDEXT    ;Save in block
        MOVSI  S1,'DSK'           ;Get structure name
        STORE  S1,GJFBLK+.FDSTR    ;Save the structure
        MOVEI  S2,[FLDDB. (.CMOFI,,,,<DFCIB.LOG>)] ;Input file
        $CALL  CMDPRS      ;Hello GLXLIB
        MOVE   S1,[GJFBLK,,LOGFD] ;Set up to copy into FD
        BLT    S1,LOGFD+GJFSIZ-1  ;Clear the block
        $CALL  CMDCFM      ;Confirm the command
        JRST   LOGOPN     ;Open the log file and return
```

```

825          SUBTTL Select Command
826
827          ;SELECT command
828
829          ;This routine is selects a node in table SNODET.
830
831          002213' 201 02 0 00 005613' .SELEC: GUIDE <node number or all> ;Noise word
832          002214' 260 17 0 00 004443'
833          002215' 201 02 0 00 001552' MOVEI S2,SETNN ;Point to parse table
834
835          002216' 260 17 0 00 004443' .SELNO: $CALL CMDPRS ;Parse that please
836          002217' 306 03 0 00 001555' CAIN T1,SNALL ;Did we get a ALL ?
837          002220' 254 00 0 00 002243' JRST .SELNA ; Yes
838          002221' 331 00 0 00 000002 SKIPL S2 ;Input negative number?
839          002222' 303 02 0 00 000017 CAILE S2,^D15 ;Skip if 15 or less
840          002223' 254 00 0 00 005630' CMDERR <Node number out of range 0-15> ;out of range
841
842          ;Check if it's our own node number
843
844          002224' 260 17 0 00 000000* $CALL LNODE ;Get our node number
845          002225' 316 02 0 00 000000* CAMN S2,LNODEN ;Is it our node number ?
846          002226' 254 00 0 00 005655' CMDERR <You cannot select our own node number (^D/S2/)> ;out of range
847
848          ;Validate that entry in SNODET by writting it to -1
849
850          002227' 202 02 0 00 000362' .SELN1: MOVEM S2,NODEN ;Put in NODEN for parse command usage
851          002230' 332 00 0 02 001436' SKIPE SNODET(S2) ;Was this node selected?
852          002231' 254 00 0 00 005677' CMDERR <Node ^D/S2/ was already selected>
853          002232' 261 17 0 00 000002 PUSH P,S2 ;Save the node number
854          002233' 201 02 0 00 005712' MOVEI S2,[FLDDB.(.CMCMA,,, <Comma to select additional nodes>,.CFM)]
855          002234' 260 17 0 00 004443' $CALL CMDPRS ;Parse the command
856          002235' 201 02 0 00 005715' MOVEI S2,[FLDDB.(.CMNUM,,^D10,<node number, 0-15>)]
857          002236' 302 03 0 00 001540' CAIE T1,CFM ;Was last character a comma ?
858          002237' 260 17 0 00 002216' $CALL .SELNO ;Yes, get the next node number
859          002240' 262 17 0 00 000002 POP P,S2 ;Restore the node number
860          002241' 476 00 0 02 001436' SETOM SNODET(S2) ;Write it to -1
861          002242' 263 17 0 00 000000 $RET ;Return
862
863          ;Routine to select node ALL
864
865          002243' 260 17 0 00 004435' .SELNA: $CALL CMDCFM ;Confirm command
866          002244' 402 00 0 00 000001 SELNA: SETZM S1 ;First node number to try
867          002245' 402 00 0 00 000002 .SLNA1: SETZM S2 ;Don't use CID
868          002246' 260 17 0 00 000000* $CALL RCON ;Get connection data
869          002247' 254 00 0 00 002253' JRST .SLNA2 ;This node doesn't exist
870          002250' 476 00 0 01 001436' SETOM SNODET(S1) ;Validate this node to be selected
871          002251' 260 17 0 00 001753' $TEXT (,<Selecting node ^D/S1/,>)
872          002253' 350 00 0 00 000001 .SLNA2: AOS S1 ;Bump index to next node number
873          002254' 307 01 0 00 000017 CAIG S1,^D15 ;Tried all nodes ?
874          002255' 254 00 0 00 002245' JRST .SLNA1 ; No
875          002256' 263 17 0 00 000000 $RET
  
```



```

876          SUBTTL Deselect Command
877
878          ;DESELECT command
879
880          ;This routine deselects a node to test in the node table. The node address is
881          ;typed in as a decimal number. Multiple addresses can be separated by commas.
882
883          002257' 201 02 0 00 005613' .DESEL: GUIDE <node number or all> ;Noise word
884          002260' 260 17 0 00 004443'
885          002261' 201 02 0 00 001552'          MOVEI S2,SETNN ;Point to parse table
886
887          002262' 260 17 0 00 004443' .DESN0: $CALL CMDPRS ;Parse that please
888          002263' 306 03 0 00 001555'          CAIN T1,SNALL ;Did we get a ALL ?
889          002264' 254 00 0 00 002303'          JRST .DESNA ;Yes
890          002265' 331 00 0 00 000002          SKIPL S2 ;-ive number?
891          002266' 303 02 0 00 000017'          CAILE S2,^D15 ;Skip if 15 or less
892          002267' 254 00 0 00 005630'          CMDERR <Node number out of range 0-15> ;out of range
893
894          ;Invalidate node entry in table SNODET by writting it to 0
895
896          002270' 336 00 0 02 001436'          SKIPN SNODET(S2) ;Was this node selected?
897          002271' 254 00 0 00 005747'          CMDERR <Node ^D/S2/ wasn't selected> ;No
898          002272' 261 17 0 00 000002'          PUSH P,S2 ;Save offset
899          002273' 201 02 0 00 005762'          MOVEI S2,[FLDDB.(.CMCMA,,, <Comma to deselect additional nodes>,.CFM)]
900          002274' 260 17 0 00 004443'          $CALL CMDPRS ;Parse that
901          002275' 201 02 0 00 005715'          MOVEI S2,[FLDDB.(.CMNUM,,^D10,<node number, 0-15>)]
902          002276' 302 03 0 00 001540'          CAIE T1,CFM ;Was last character a comma ?
903          002277' 260 17 0 00 002262'          $CALL .DESN0 ;Yes, get the next node number
904          002300' 262 17 0 00 000002'          POP P,S2 ;Restore node number
905          002301' 402 00 0 02 001436'          SETZM SNODET(S2) ;Invalidate this entry
906          002302' 263 17 0 00 000000'          $RET ;Return
907
908          ;Deselect all nodes
909
910          002303' 260 17 0 00 004435' .DESNA: $CALL CMDCFM ;Confirm command
911          002304' 402 00 0 00 001436' DESNA: SETZM SNODET ;Clear node 0
912          002305' 200 01 0 00 005765'          MOVE S1,[SNODET,SNODET+1] ;Set up BLT pointer
913          002306' 251 01 0 00 001455'          BLT S1,SNODET+^D15 ;Zap
914          002307' 263 17 0 00 000000'          $RET ;Off we go

```

```

915          SUBTTL  HELP Command
916
917          ;Here on a HELP command
918
919 002310' 200 06 0 00 005766' .HELP:  MOVE  T4,[POINT 7,HLPBUF] ;Get pointer to help buffer
920 002311' 403 05 0 00 001230'      SETZB  T3,HLPBUF ;Make easy check for nothing typed
921 002312' 201 02 0 00 001652'      MOVEI  S2,HLPARS ;Help table
922 002313' 260 17 0 00 004443'      $CALL  CMDPRS ;Parse it
923 002314' 260 17 0 00 002437'      $CALL  HLPATM ;Copy that atom please
924 002315' 254 00 0 00 002332'      JRST   HELP1 ;It was confirmed, continue
925 002316' 621 02 0 00 777777'      TLZ    S2,-1 ;Clear left half (address of ASCIIZ)
926 002317' 205 03 0 00 777775'      MOVSI  T1,-HLPADL ;Load AOBJN pointer to additional help
927 002320' 554 04 0 03 001666' HELPO: HLRZ  T2,HLPADD(T1) ;Load additional parse table addr
928 002321' 312 02 0 00 000004'      CAME  S2,T2 ;Match something we know more about?
929 002322' 253 03 0 00 002320'      AOBJN  T1,HELPO ;No, loop
930 002323' 550 02 0 03 001666'      HRRZ  S2,HLPADD(T1) ;Point to more stuff
931 002324' 260 17 0 00 004443'      $CALL  CMDPRS ;Parse that
932 002325' 260 17 0 00 002437'      $CALL  HLPATM ;Copy the atom
933 002326' 254 00 0 00 002332'      JRST   HELP1 ;It was confirmed
934 002327' 260 17 0 00 004435'      $CALL  CMDCFM ;We expect command confirmation here
935 002330' 201 01 0 00 000000'      MOVEI  S1,0 ;Load a zero
936 002331' 137 01 0 00 000006'      DPB    S1,T4 ;Insure a zero there
937
938          ;Command has been parsed and thing to get help on is in HLPBUF, set up flags
939
940 002332' 402 00 0 00 001363' HELP1: SETZM  HLPEOF ;Not EOF on help file
941 002333' 402 00 0 00 000006'      SETZM  T4 ;Initially not outputting anything
942 002334' 200 01 0 00 001230'      MOVE  S1,HLPBUF ;Load help buffer contents
943 002335' 312 01 0 00 005157'      CAME  S1,[ASCIIZ/*/] ;Was it star?
944 002336' 336 00 0 00 001230'      SKIPN  HLPBUF ;Skip if something typed
945 002337' 477 05 0 00 000006'      SETOB  T3,T4 ;Yes, output first text you see
946
947          ;Open up the help file for reading.
948
949 002340' 120 01 0 00 005767'      DMOVE  S1,[EXP <SIXBIT/DSK/>,<ASCII/dsk:d/>] ;Load first place to look
950 002341' 260 17 0 00 002462'      $CALL  HFIND ;Try and find it there
951 002342' 326 00 0 00 002351'      JUMPT  HELP2 ;Jump if we got it
952 002343' 120 01 0 00 005771'      DMOVE  S1,[EXP <SIXBIT/HLP/>,<ASCII/hlp:d/>] ;Load second place
953 002344' 260 17 0 00 002462'      $CALL  HFIND ;Try and find it there
954 002345' 326 00 0 00 002351'      JUMPT  HELP2 ;Jump if we got it
955 002346' 120 01 0 00 005773'      DMOVE  S1,[EXP <SIXBIT/SYS/>,<ASCII/sys:d/>] ;Load third place to look
956 002347' 260 17 0 00 002462'      $CALL  HFIND ;Try and find it there
957 002350' 322 00 0 00 002474'      JUMPF  HLPERR ;Jump if we didn't get it

```


958
 959
 960
 961 002351' 332 00 0 00 001363'
 962 002352' 254 00 0 00 002375'
 963 002353' 260 17 0 00 002402'
 964
 965
 966
 967 002354' 302 03 0 00 000052
 968 002355' 254 00 0 00 002371'
 969 002356' 402 00 0 00 000006
 970 002357' 336 00 0 00 001230'
 971 002360' 476 00 0 00 001363'
 972 002361' 561 02 0 00 001330'
 973 002362' 561 01 0 00 001230'
 974 002363' 260 17 0 00 000000*
 975 002364' 200 02 0 00 001230'
 976 002365' 312 02 0 00 005157'
 977 002366' 336 00 0 00 000001
 978 002367' 477 05 0 00 000006
 979 002370' 254 00 0 00 002351'
 980
 981
 982
 983 002371' 561 01 0 00 001330'
 984 002372' 332 00 0 00 000006
 985 002373' 260 17 0 00 004350'
 986 002374' 254 00 0 00 002351'
 987
 988
 989
 990 002375' 336 00 0 00 000005
 991 002376' 260 17 0 00 002251*
 992 002400' 200 01 0 00 001364'
 993 002401' 324 17 0 00 001721*

;Ready to read help file, line at a time. First check EOF flag.

HELP2: SKIPE HLPEOF ;Eof?
 JRST HELP7 ;Release it and return to top level
 \$CALL HREAD ;Read in a line from the file

;Check the first character on the line, if its a '*' then we must check more

HELP3: CAIE T1,'*' ;Is it a keyword start?
 JRST HELP6 ;Nope
 SETZM T4 ;Not outputting text any more
 SKIPN HLPBUF ;Skip if some specific help request
 SETOM HLPEOF ;Yes, simulate EOF at first '*'
 HRROI S2,HLPLIN ;Point to help line
 HRROI S1,HLPBUF ;Point to buffer
 \$CALL S%SCMP ;Call string comparison routine
 MOVE S2,HLPBUF ;Load help buffer contents
 CAME S2,[ASCIZ/*/] ;Was it star?
 SKIPN S1 ;Exact match?
 SETOB T3,T4 ;Yes, we should output text now
 JRST HELP2 ;Get another line

;Here if line didn't begin with '*', output the line if output flag is set.

HELP6: HRROI S1,HLPLIN ;Point to line of text
 SKIPE T4 ;Outputting now?
 \$CALL SOUT%% ;Output to terminal
 JRST HELP2 ;Loop for more

;Here at end of file, determine if something has been printed.

HELP7: SKIPN T3 ;Skip if something was found
 \$TEXT (,<% Sorry, no information on '%T/HLPBUF/'>)
 MOVE S1,HLPBUF ;Load the help IFN
 PJRST F%REL ;Release it and return

```

994
995
996
997 002402' 200 01 0 00 001364'
998 002403' 200 04 0 00 006013'
999 002404' 260 17 0 00 000000*
1000 002405' 322 00 0 00 002415'
1001 002406' 302 02 0 00 000041
1002 002407' 334 03 0 00 000002
1003 002410' 474 03 0 00 000000
1004 002411' 302 02 0 00 000052
1005 002412' 136 02 0 00 000004
1006
1007 002413' 260 17 0 00 002404*
1008 002414' 326 00 0 00 002417'
1009
1010 002415' 476 00 0 00 001363'
1011 002416' 254 00 0 00 002434'
1012
1013 002417' 306 02 0 00 000015
1014 002420' 254 00 0 00 002413'
1015 002421' 306 02 0 00 000012
1016 002422' 254 00 0 00 002425'
1017 002423' 136 02 0 00 000004
1018 002424' 254 00 0 00 002413'
1019
1020 002425' 321 03 0 00 002402'
1021 002426' 306 03 0 00 000052
1022 002427' 254 00 0 00 002434'
1023 002430' 201 02 0 00 000015
1024 002431' 136 02 0 00 000004
1025 002432' 201 02 0 00 000012
1026 002433' 136 02 0 00 000004
1027
1028 002434' 400 02 0 00 000000
1029 002435' 136 02 0 00 000004
1030 002436' 263 17 0 00 000000

;Here to read a line from the help file into HLPLIN, returns S1/first character
HREAD:  MOVE S1,HLPIFN      ;Load the IFN
        MOVE T2,[Point 7,HLPLIN] ;Load pointer to line of text
        $CALL F%IBYT        ;Get the first character of the line
        JUMPF HREAD2        ;Punt - EOF
        CAIE S2,'*'         ;Eat this line?
        SKIPA T1,S2         ;No, copy the character to T1
        SETO T1             ;Indicate it was a comment line
        CAIE S2,'*'         ;Asterisk?
        IDPB S2,T2          ;No, store the 1st character of line

HREAD1: $CALL F%IBYT        ;Get a byte
        JUMPT HREAD3        ;Jump if not EOF

HREAD2: SETOM HLPEOF        ;Now it was an EOF
        JRST HREAD5         ;Deposit a null and return

HREAD3: CAIN S2,.CHCRT      ;Was it a return?
        JRST HREAD1         ;Yes, eat it
        CAIN S2,.CHLFD      ;Was it a linefeed?
        JRST HREAD4         ;Yes, end of line
        IDPB S2,T2          ;Store the character
        JRST HREAD1         ;Loop

HREAD4: JUMPL T1,HREAD      ;If it was a comment start another line
        CAIN T1,'*'         ;Line start with a star?
        JRST HREAD5         ;Yes
        MOVEI S2,.CHCRT      ;Load a return
        IDPB S2,T2          ; and store it
        MOVEI S2,.CHLFD      ;Load a line feed
        IDPB S2,T2          ; and store that

HREAD5: SETZ S2             ;Clear S2
        IDPB S2,T2          ;Store it here
        $RET                ;Return

```

```

1031
1032 ;Copy atom buffer to help text buffer, returns T1/PDB used
1033
1034 002437' 306 03 0 00 001540' HLPATM: CAIN T1,CFM ;Was it a confirm?
1035 002440' 254 00 0 00 002454' JRST HLPAT3 ;Yes
1036 002441' 306 03 0 00 001663' CAIN T1,HAL ;Was it help all?
1037 002442' 254 00 0 00 002457' JRST HLPAT4 ;yes
1038 002443' 554 01 0 02 000000' HLRZ S1,(S2) ;Get address of ASCIZ text
1039 002444' 505 01 0 00 440700' HRLI S1,(Point 7,) ;Make byte pointer
1040
1041 002445' 134 03 0 00 000001' HLPAT1: ILDB T1,S1 ;Get first character
1042 002446' 306 03 0 00 000000' CAIN T1,0 ;Loading a zero?
1043 002447' 201 03 0 00 000040' MOVEI T1,' ' ;Make it a space
1044 002450' 136 03 0 00 000006' IDPB T1,T4 ;Not zero, put it in help buffer
1045 002451' 302 03 0 00 000040' CAIE T1,' ' ;Space?
1046 002452' 254 00 0 00 002445' JRST HLPAT1 ;No, loop for more
1047 002453' 254 00 0 00 000000* JRST .POPJ1 ;Skip return
1048
1049 002454' 400 01 0 00 000000' HLPAT3: SETZ S1, ;Load a zero
1050 002455' 137 01 0 00 000006' DPB S1,T4 ;Tie off the request
1051 002456' 263 17 0 00 000000' $RET ;Return
1052
1053 002457' 205 01 0 00 250000' HLPAT4: MOVSI S1,(<ASCIZ/*/>) ;Load a star
1054 002460' 202 01 0 00 001230' MOVEM S1,HLPBUF ;Save it as the request type
1055 002461' 254 00 0 00 004435' JRST CMDCFM ;Confirm it and return
1056
1057 ;Here to try and find the help file, called with S1/TOPS-10 dev, S2/TOPS-20 dev
1058 ;Returns false if not found, returns true with file open and HLPIFN set.
1059
1060 002462' 202 01 0 00 000243' HFIND: MOVEM S1,HLPFD1+.FDSTR ;Save TOPS-10 structure name
1061 002463' 202 02 0 00 000232' MOVEM S2,HLPFD2+.FDSTG ;Save TOPS-20 first 5 characters
1062 002464' 332 00 0 0C 000365' SKIPE MONTYP ;Skip if TOPS-10
1063 002465' 334 02 0 00 006014' SKIPA S2,[HFOB20] ;TOPS-20
1064 002466' 201 02 0 00 000235' MOVEI S2,HFOB10 ;TOPS-10
1065 002467' 201 01 0 00 000005' MOVEI S1,FOB.SZ ;Load size of the FOB
1066 002470' 260 17 0 00 000000* $CALL F%IOPN ;Open up input file
1067 002471' 322000 000000* $RETIFF ;Return if false
1068 002472' 202 01 0 00 001364' MOVEM S1,HLPIFN ;Save IFN of help file
1069 002473' 324740 000000* $RETT ;Return OK
1070
1071 ;Here if help file open problem
1072
1073 002474' 260 17 0 00 002376* HLPERR: $TEXT (,<? Help not available - ^E/S1/>)
1074 002476' 263 17 0 00 000000 $RET ;Return

```

```

1075          SUBTTL  RUN COMMAND
1076
1077          ;Here we have the run command, the format is:
1078          ;      DFCIB> RUN/PASS:n T1/ITERATION:n,T2/ITERATION:n(CRLF)
1079          ;or      DFCIB> RUN T1/ITERATION:n,T2/ITERATION:n(CRLF)
1080          ;These command strings create dispatch a table in the run buffer, the run
1081          ;buffer may be re-executed in the following manner:
1082          ;      RUN/PASS:n(CRLF)
1083          ;or      RUN(CRLF)
1084          ;The test and iteration counts will be performed from the last 'RUN' command
1085          ;that updated the run buffer.  If the run buffer is empty, the parser will
1086          ;report 'NO TESTS SPECIFIED - DISPATCH TABLE EMPTY'.
1087
1088          .RUN:  GUIDE  <test or script>          ;Help noise word
1089
1090          SETZM  PASCNT          ;Clear the pass count reporter
1091          SETZM  PASSCN          ;Clear pass counter
1092          AOS    PASSCN          ;Set pass counter to one
1093          MOVEI  S2,RPPARS       ;Point to Run Parse table
1094          $CALL  CMDPRS          ;Parse that please
1095          CAIN   T1,CFM          ;Did we get command confirmation?
1096          JRST   RLAST          ;Yes, run last dispatch table
1097          ;      using pass count and old iterations
1098          CAIE   T1,RPPARS       ;No, did we get a switch?
1099          JRST   RUN1           ;Must be a test, make table entry
1100
1101          ;Here if RUN /PASSES:, input the pass count.
1102
1103          MOVEI  S2,[FLDDB.(.CMNUM,,^D10,<pass count, 1-131071>)]
1104          $CALL  CMDPRS          ;Parse the number
1105          TDNE   S2,[-1,,1B18]   ;Is number in range?
1106          CMDERR <Pass count out of range> ;No, print error message
1107          MOVEM  S2,PASSCN       ;Save pass count
1108
1109          ;Got the value to /PASSES, look for a test, script, or confirm.
1110
1111          MOVEI  S2,RUPARS       ;Parse the rest of the run options
1112          $CALL  CMDPRS          ;Like the test names, etc
1113          CAIN   T1,CFM          ;Did we get command confirmation?
1114          JRST   RLAST          ;Yes, now run test in dispatch buffer
1115          ;      using pass count and old iterations

```



```

1116
1117
1118
1119 002523' 402 00 0 00 001365'
1120 002524' 200 01 0 00 006064'
1121 002525' 251 01 0 00 001434'
1122 002526' 201 07 0 00 001365'
1123 002527' 260 17 0 00 002573'
1124
1125
1126
1127 002530' 201 02 0 00 001641'
1128 002531' 260 17 0 00 004443'
1129 002532' 306 03 0 00 001540'
1130 002533' 254 00 0 00 002605'
1131 002534' 302 03 0 00 001644'
1132 002535' 254 00 0 00 002550'
1133
1134
1135
1136 002536' 201 02 0 00 006073'
1137 002537' 260 17 0 00 004443'
1138 002540' 612 02 0 00 006045'
1139 002541' 254 00 0 00 006111'
1140 002542' 540 06 0 00 000002'
1141 002543' 202 06 0 07 000000'
1142
1143
1144
1145 002544' 201 02 0 00 006124'
1146 002545' 260 17 0 00 004443'
1147 002546' 306 03 0 00 001540'
1148 002547' 254 00 0 00 002605'
  
```

;Here if the first test name in the command was typed, copy it to the run table

```

RUN1:  SETZM  MAINBF      ;Clear first word of run buffer
        MOVE  S1,[MAINBF,,MAINBF+1] ;Prepare BLT pointer
        BLT   S1,MAINBF+MAINBL-1    ;Zap
        MOVEI P1,MAINBF      ;Get pointer to run buffer
        $CALL RDEFIC        ;Go set default iteration count
  
```

;Now lets see if user wants to specify iteration count, another test or CRLF

```

        MOVEI S2,RVPARS      ;Parse for comma, /ITER, or confirm
        $CALL CMDPRS        ;Parse that command
        CAIN  T1,CFM        ;Was parse a carriage return ?
        JRST  RUNTST        ;Yes, go do the testing
        CAIE  T1,ISPARS     ;Was parse a switch ?
        JRST  RCOMMA        ;No, it must be a comma
  
```

;Here if /ITERATIONS, read the number then continue parsing

```

RUN4:  MOVEI  S2,[FLDDB.(.CMNUM,,^D10,<iteration count, 1-131071>)]
        $CALL CMDPRS        ;Parse the command
        TDNE  S2,[-1,,1B18] ;Is number in range ?
        CMDERR <iteration count out of range> ;No, print error message
        HRR   T4,S2         ;Add iteration count to test
        MOVEM T4,(P1)       ;Put into run buffer
  
```

;Here after test name and /ITERATIONS input, get either a comma or a confirm.

```

RUN2:  MOVEI  S2,[FLDDB.(.CMCMA,,,<Comma to enter additional tests>,,CFM)]
        $CALL CMDPRS        ;Parse that please
        CAIN  T1,CFM        ;Was it a CRLF ?
        JRST  RUNTST        ;Yes, go run testing
  
```



```

1149
1150 ;Here when comma typed, point to next run list entry then get a test name.
1151
1152 002550' 271 07 0 00 000002' RCOMMA: ADDI P1,2 ;Bump test counter two locations
1153 002551' 303 07 0 00 001434' CAILE P1,MAINBF+MAINBL-1 ;Have we exceeded 20 tests, at two
1154 ; locations for each test ?
1155 002552' 254 00 0 00 006147' CMDERR <Exceeded test entry quota, maximum of 20 entries allowed>
1156
1157 002553' 201 02 0 00 001623' MOVEI S2,RVPARS ;Parse another test/script
1158 002554' 260 17 0 00 004443' $CALL CMDPRS ;Parse that one
1159 002555' 260 17 0 00 002573' $CALL RDEFIC ;Go setup default iteration count
1160
1161 ;Here after 2nd-Nth test name input, accept either a comma or confirm to go.
1162
1163 002556' 201 02 0 00 001641' MOVEI S2,RVPARS ;Parse using macro at tag
1164 002557' 260 17 0 00 004443' $CALL CMDPRS ;Parse that one
1165 002560' 306 03 0 00 001540' CAIN T1,CFM ;Was parse a carriage return ?
1166 002561' 254 00 0 00 002605' JRST RUNTST ;Yes, go do the testing
1167 002562' 302 03 0 00 001644' CAIE T1,ISPARS ;Was parse a /ITERATION switch ?
1168 002563' 254 00 0 00 002550' JRST RCOMMA ;No, it must be a comma
1169 002564' 254 00 0 00 002536' JRST RUN4 ;Yes, input value
1170
1171 ;Here when command confirmed without a test specified, make sure that we are
1172 ; not running an empty test table.
1173
1174 002565' 201 01 0 00 001365' RLAST: MOVEI S1,MAINBF ;Get pointer to test table
1175 002566' 554 02 0 01 000000 HLRZ S2,(S1) ;Get first test
1176 002567' 326 02 0 00 002605' JUMPN S2,RUNTST ;Is there a table entry ?
1177 002570' 260 17 0 00 002474* $TEXT (,<? Run table is empty>) ;Yes, report table empty
1178 002572' 263 17 0 00 000000 $RET ;Return to top level parser
1179
1180 ;Here we setup default iteration count, and save the test name for
1181 ; SHOW RUN-TABLE command
1182
1183 002573' 514 06 0 02 000000 RDEFIC: HRLZ T4,(S2) ;Get test tag
1184 002574' 550 01 0 02 000000 HRRZ S1,(S2) ;Load test address
1185 002575' 302 01 0 00 001611* CAIE S1,TST99 ;Running test 99 or
1186 002576' 306 01 0 00 001621* CAIN S1,XRCSE ; the exerciser script?
1187 002577' 664 06 0 00 000144 TROA T4,^D100 ;yes, set default iterations to 100
1188 002600' 660 06 0 00 000001 TRO T4,1 ;Set default iteration count = 1
1189 002601' 202 06 0 07 000000 MOVEM T4,(P1) ;Put it in run buffer
1190 002602' 554 10 0 02 000000 HLRZ P2,(S2) ;Get pointer to test ascii name
1191 002603' 202 10 0 07 000001 MOVEM P2,1(P1) ;Put test name away for show run-table
1192 002604' 263 17 0 00 000000 $RET ;Return to call

```

```

1193
1194 ;Here after confirm typed to actually run the tests in the run table.
1195 ;For each pass, for each test, for each iteration, for each node, run a test
1196 ;ACs:
1197 ;      P1/ pointer into the run table (MAINBF)
1198 ;      P2/ routine to call from the run table
1199
1200 002605' 260 17 0 00 002642' RUNTST: $CALL LOCKIT ;Lock in core
1201 002606' 350 00 0 00 000355' RUNTS1: AOS PASCNT ;Count this pass
1202 002607' 201 07 0 00 001365' MOVEI P1,MAINBF ;Get pointer to test table
1203
1204 ;Here to run the next test of this pass, set test iteration count.
1205
1206 002610' 550 01 0 07 000000 RUNTS2: HRRZ S1,(P1) ;Get the iteration count
1207 002611' 202 01 0 00 000360' MOVEM S1,TITCNT ;Save in iteration count
1208 002612' 554 10 0 07 000000 HLRZ P2,(P1) ;Get test to run
1209 002613' 326 10 0 00 002621' JUMPN P2,RUNTSA ;Jump if not end of table
1210
1211 ;End of run table, count down passes and loop if not done.
1212
1213 002614' 200 01 0 00 000355' MOVE S1,PASCNT ;Load pass count
1214 002615' 312 01 0 00 000356' CAME S1,PASSCN ;Done all passes?
1215 002616' 254 00 0 00 002606' JRST RUNTS1 ;Not done, continue
1216 002617' 260 17 0 00 002652' $CALL UNLKIT ;Unlock
1217 002620' 263 17 0 00 000000 $RET ;Finished with all passes
1218
1219 ;Here to run the test, iteration count is set up
1220
1221 002621' 260 17 0 00 004257' RUNTSA: $CALL GETFN ;Select first node
1222 002622' 263 17 0 00 000000 $RET ;There weren't any nodes selected
1223
1224 ;Node has been selected, call test in question
1225
1226 002623' 402 00 0 00 000357' RNTSA1: SETZM ITCNT ;Clear iteration count
1227
1228 002624' 350 00 0 00 000357' RNTSA2: AOS ITCNT ;Count up printable iteration count
1229 002625' 260 17 0 00 006164' $CALL [SAVE <P1,P2,P3> ;Save ACs important to us
1230 002626' 260 17 0 00 004541' JRST (P2)] ;Run the test
1231 002627' 200 01 0 00 000357' $CALL RTCHK ;Check runtime status
1232 002628' 312 01 0 00 000360' MOVE S1,ITCNT ;Load iteration count
1233 002629' 254 00 0 00 002624' CAME S1,TITCNT ;All done with this iteration?
1234 002631' 254 00 0 00 002624' JRST RNTSA2 ;No, run another iteration
1235
1236 002632' 302 10 0 00 002575* CAIE P2,TST99 ;Is this
1237 002633' 306 10 0 00 002576* CAIN P2,XRCSE ; the exerciser?
1238 002634' 254 00 0 00 002640' JRST RNTSA3 ;Yes, it selects its own nodes
1239
1240 002635' 260 17 0 00 004277' $CALL GETNN ;See if there are more nodes selected
1241 002636' 254 00 0 00 002640' JRST RNTSA3 ;No more nodes
1242 002637' 254 00 0 00 002623' JRST RNTSA1 ;There are more nodes
1243
1244 002640' 271 07 0 00 000002 RNTSA3: ADDI P1,2 ;Bump test pointer two locations
1245 002641' 254 00 0 00 002610' JRST RUNTS2 ;Go run next test

```

```
1246
1247 ;Here to lock low seg in core
1248
1249 002642' 332 00 0 00 000365' LOCKIT: SKIPE MONTYP ;TOPS-10?
1250 002643' 263 17 0 00 000000 $RET ;Nope
1251 002644' 201 01 0 00 000017 MOVEI S1,LK.LLC!LK.LNP!LK.LNE!LK.LLS ;Lock low seg, cache, non cont
1252 002645' 047 01 0 00 000060 LOCK S1, ;or CALLI AC,60
1253 002646' 260 17 0 00 002570* $TEXT (.<? LOCK UUO failed, error code ^0/S1/>)
1254 002650' 476 00 0 00 000373' SETOM LCKFLG ;We are locked now
1255 002651' 263 17 0 00 000000 $RET ;Return
1256
1257 ;Here to unlock from core
1258
1259 002652' 336 00 0 00 000365' UNLKIT: SKIPN MONTYP ;TOPS-20?
1260 002653' 336 00 0 00 000373' SKIPN LCKFLG ;Locked?
1261 002654' 263 17 0 00 000000 $RET ;Nope
1262 002655' 201 01 0 00 000001 MOVEI S1,1 ;Low segment
1263 002656' 047 01 0 00 000120 UNLOK S1, ;or CALLI AC,120
1264 002657' 260 17 0 00 002646* $TEXT (.<% UNLOK UUO failed>) ;Punt
1265 002661' 402 00 0 00 000373' SETZM LCKFLG ;No longer locked
1266 002662' 263 17 0 00 000000 $RET ;Return
```



```

1267          SUBTTL Show Command
1268
1269          ;SHOW command
1270
1271 002663' 201 02 0 00 006215' .SHOW: GUIDE <state of> ;Noise word for SHOW command
1272 002664' 260 17 0 00 004443'
1273 002665' 201 02 0 00 006217' MOVEI S2,[FLDDB.(.CMKEY,,SHOTAB)] ;Parse table
1274 002666' 260 17 0 00 004443' $CALL CMDPRS ;Parse that please
1275 002667' 550 01 0 02 000000 HRRZ S1,(S2) ;Provide routine address subject
1276 002670' 324 17 0 01 000000 PJRST (S1) ;Go provide SHOW on subject
1277
1278          ;SHOW ALL-PROGRAM-PARAMETERS
1279
1280 002671' 260 17 0 00 004435' SHALL: $CALL CMDCFM ;Confirm that command please
1281 002672' 260 17 0 00 002733' $CALL .SNODE ;Show local node number
1282 002673' 260 17 0 00 002702' $CALL .SHSPS ;Show selected paths
1283 002674' 260 17 0 00 002736' $CALL .SHSN ;Routine to show selected node
1284 002675' 260 17 0 00 002755' $CALL .STIOU ;Show time out
1285 002676' 260 17 0 00 002770' $CALL .SRTBL ;Show entries in run table
1286 002677' 260 17 0 00 002713' $CALL .SHSWI ;Show switches
1287 002700' 263 17 0 00 000000 $RET ;Return, continue parse
1288
1289          ;SHOW PATH-SELECTION
1290
1291 002701' 260 17 0 00 004435' SHSPS: $CALL CMDCFM ;Confirm that
1292 002702' 550 01 0 00 002135' .SHSPS: HRRZ S1,SL3B ;Load the number of keywords
1293 002703' 550 02 0 01 002135' SHSPS1: HRRZ S2,SL3B(S1) ;Load a path specifier
1294 002704' 312 02 0 00 000361' CAME S2,PATHS ;Does it match?
1295 002705' 367 01 0 00 002703' SCJG S1,SHSPS1 ;No, loop
1296 002706' 554 02 0 01 002135' HLIZ S2,SL3B(S1) ;Point to the path text
1297 002707' 260 17 0 00 002657* $TEXT (,<Path selection is ^T/(S2)/>)
1298 002711' 263 17 0 00 000000 $RET ;Return
1299
1300          ;SHOW SWITCHES
1301
1302 002712' 260 17 0 00 004435' SHSWI: $CALL CMDCFM ;Confirm that please
1303 002713' 260 17 0 00 002707* .SHSWI: $TEXT (,<Switches set:^A>) ;Label the set switches
1304 002715' 205 05 0 00 332006 MOVSI T3,(<SKIPE (T4)>) ;Load switches that are set
1305 002716' 260 17 0 00 002722' $CALL SHSW1 ;Go print them
1306 002717' 260 17 0 00 002713* $TEXT (,<Switches clear:^A>) ;Label the clear switches
1307 002721' 205 05 0 00 336006 MOVSI T3,(<SKIPN (T4)>) ;Load instr
1308
1309 002722' 205 04 0 00 777774 SHSW1: MOVSI T2,-SWITCN ;Set up number of switches
1310 002723' 550 06 0 04 001546' SHSW2: HRRZ T4,SWITAB+1(T2) ;Load the mask bit
1311 002724' 554 03 0 04 001546' HLRZ T1,SWITAB+1(T2) ;Load the ASCII string address
1312 002725' 256 00 0 00 000005 XCT T3 ;Skip whatever
1313 002726' 260 17 0 00 002717* $TEXT (,< ^T/(T1)/^A>) ;Yes, print it
1314 002730' 253 04 0 00 002723' AOBJN T2,SHSW2 ;Bump counter, loop if not done
1315 002731' 324 17 0 00 004343' PJRST PCRLF ;Output CRLF and return

```

```

1316
1317
1318
1319 002732' 260 17 0 00 004435'
1320 002733' 260 17 0 00 000000*
1321 002734' 263 17 0 00 000000
1322
1323
1324
1325 002735' 260 17 0 00 004435'
1326 002736' 476 00 0 00 000002
1327 002737' 205 01 0 00 777760
1328 002740' 336 00 0 01 001436'
1329 002741' 254 00 0 00 002747'
1330 002742' 356 00 0 00 000002
1331 002743' 260 17 0 00 002726*
1332 002745' 260 17 0 00 002743*
1333 002747' 253 01 0 00 002740'
1334 002750' 335 00 0 00 000002
1335 002751' 260 17 0 00 002745*
1336 002753' 324 17 0 00 004343'
1337
1338
1339
1340 002754' 260 17 0 00 004435'
1341 002755' 304 00 0 00 000000
1342 002762' 200 01 0 00 002126'
1343 002763' 231 01 0 00 001750
1344 002764' 260 17 0 00 002751*
1345 002766' 263 17 0 00 000000
1346
1347
1348
1349 002767' 260 17 0 00 004435'
1350 002770' 201 02 0 00 001365'
1351 002771' 554 03 0 02 000000
1352 002772' 322 03 0 00 003003'
1353
1354 002773' 260 17 0 00 002764*
1355
1356 002775' 554 03 0 02 000000
1357 002776' 322 03 0 00 002471*
1358 002777' 260 17 0 00 002773*
1359 003001' 271 02 0 00 000001
1360 003002' 344 02 0 00 002775'
1361
1362 003003' 260 17 0 00 002777*
1363 003005' 263 17 0 00 000000

;SHOW LOCAL-NODE-NUMBER
SLNODE: $CALL CMDCFM ;Confirm command
.SNODE: $CALL GLNN ;DFCIB1.MAC
$RET

;SHOW SELECTED-NODES
SHSN: $CALL CMDCFM ;Confirm command
.SHSN: SETOM S2 ;Clear indicator
MOVSI S1,-^D16 ;Initialize pointer
SHSN1: SKIPN SNODET(S1) ;Is this node selected ?
JRST SHSN2 ;No
AOSN S2 ;First time through here?
$TEXT (,<Selected nodes: ^A>) ;Yep
$TEXT (,<^D/S1,RHMASK/ ^A>) ;Output the number space space
SHSN2: AOBJN S1,SHSN1 ;Loop for all of them
SKIPGE S2 ;Any nodes displayed
$TEXT (,<No nodes have been selected^A>) ;Nope
PJRTS PCRLF ;Output CRLF and return

;SHOW TIME-OUT
SHTIOU: $CALL CMDCFM ;Confirm the command
.STIOU: $SAVE <S1,S2> ;Save a couple
MOVE S1,SECS ;Load milliseconds
IDIVI S1,^D1000 ;Compute seconds
$TEXT (,<Time out is set to ^D/S1/. seconds>)
$RET ;Return to toplevel

;Here for SHOW RUN-TABLE
SHRTBL: $CALL CMDCFM ;Confirm the command
.SRTBL: MOVEI S2,MAINBF ;Get pointer to main buffer
HLRZ T1,(S2) ;Load first test name
JUMPE T1,STRB3 ;Jump if table empty
$TEXT (,<Contents of the Run Table
Iterations Test Name>)
SRTB2: HLRZ T1,(S2) ;Get test name
JUMPE T1,.POPJ ;Return if end of table
$TEXT (,<^D9R /(S2),RHMASK/. ^T/a1(S2)/>)
ADDI S2,1 ;Bump by one
AOJA S2,SRTB2 ;Loop for next entry
STRB3: $TEXT (,<Run Table is empty>) ;Its empty
$RET ;Return

```



```

1364
1365 ;SHOW CI-CONFIGURATION
1366
1367 003006' 260 17 0 00 004435' SHCICO: $CALL CMDCFM ;Confirm that
1368 003007' 260 17 0 00 000000* $CALL PALLN ;DFCIB1.MAC
1369 003010' 263 17 0 00 000000 $RET
1370
1371 ;SHOW ID-OF-CONNECTED-NODE
1372
1373 003011' 260 17 0 00 004435' SHCID: $CALL CMDCFM ;Confirm the command
1374 003012' 260 17 0 00 003003* .SCID: $TEXT (,<The connect id is ^D/CONID/>)
1375 003014' 263 17 0 00 000000 $RET
1376
1377 ;SHOW MAPPED-BUFFER-NAME
1378
1379 003015' 260 17 0 00 004435' SHMBN: $CALL CMDCFM ;Confirm the command
1380 003016' 336 00 0 00 000000* SKIPN BUFRNM ;Skip if a buffer was mapped
1381 003017' 260 17 0 00 003012* $TEXT (,<No buffer mapped>)
1382 003021' 332 00 0 00 003016* SKIPE BUFRNM ;Skip if no buffer mapped
1383 003022' 260 17 0 00 003017* $TEXT (,<Returned mapped buffer name ^O/BUFRNM/>)
1384 003024' 263 17 0 00 000000 $RET
1385
1386 ;SHOW MINIMUM-BUFFER-SIZES
1387
1388 003025' 260 17 0 00 004435' SHMBS: $CALL CMDCFM ;Confirm the command
1389 003026' 260 17 0 00 000000* $CALL RBSIZ ;Go read minimum buffer sizes
1390 $TEXT (,<Minimum message buffer size is ^D/MINMS/. words
1391 003027' 260 17 0 00 003022* Minimum datagram buffer size is ^D/MINDS/. words>)
1392 003031' 263 17 0 00 000000 $RET
1393
1394 ;SHOW COUNTERS
1395
1396 003032' 260 17 0 00 004435' SHCO: $CALL CMDCFM ;Confirm the command
1397 003033' 260 17 0 00 000000* $CALL SCOUNT ;Do the work
1398 003034' 263 17 0 00 000000 $RET
1399
1400 ;SHOW STATUS-OF-CONNECTED-NODE
1401
1402 003035' 260 17 0 00 004435' SHSOCN: $CALL CMDCFM ;Confirm the command
1403 003036' 260 17 0 00 000000* .SSOCN: $CALL SHCST ;DFCIB1.MAC
1404 003037' 263 17 0 00 000000 $RET

```

```

1405
1406      ;SHOW PATH-STATUS-OF-NODE
1407
1408      SHPS:  GUIDE    <node number>          ;Noise word
1409      MOVEI    S2,[FLDDB.(.CMNUM,,^D10,<node number, 0-15>)]
1410      $CALL    CMDPRS          ;Parse the field please
1411      CAILE    S2,^D15          ;Skip if 15 or less
1412      CMDERR   <Node number out of range 0-15> ;Out of range
1413      MOVE     S1,S2            ;Copy the node number to S1
1414      $CALL    GPRS            ;Go get return path status for node
1415      SKIPE    SCSERS          ;Skip if SCS failure
1416      JRST     .SPS1           ;SCS failure occurred
1417      HLRZ     S1,GPRSB+.SQRPS ;Get path A status
1418      SKIPE    S1              ;nonzero means path bad
1419      SKIPA    S1,[ASCIZ/good/] ;Good
1420      MOVE     S1,[ASCIZ/bad/]  ;Bad
1421      HRRZ     S2,GPRSB+.SQRPS ;Get path B status
1422      SKIPE    S2              ;Skip if non zero (path bad)
1423      SKIPA    S2,[ASCIZ/good/] ;Good
1424      MOVE     S2,[ASCIZ/bad/]  ;Bad
1425      $TEXT    (,<Node ^D/GPRSB+.SQRPN/ Path A status is marked ^T/S1/
1426      Node ^D/GPRSB+.SQRPN/ Path B status is marked ^T/S2/>)
1427      .SPS1:  $RET              ;Return to parser
1428
1429      ;SHOW POLL-STATUS-OF-CONNECTED-NODE
1430
1431      SHPS:  $CALL    CMDCFM          ;Confirm that
1432      .SPST:  $CALL    SHPOLL         ;DFCIB1.MAC
1433      $RET
1434
1435      ;SHOW EVENT-QUEUE
1436
1437      SHEVNT: $CALL    CMDCFM          ;Confirm the command
1438      $CALL    SEVNT                ;DFCIB1.MAC
1439      $RET
1440
1408 003040' 201 02 0 00 006476'
1409 003041' 260 17 0 00 004443'
1410 003042' 201 02 0 00 005715'
1411 003043' 260 17 0 00 004443'
1412 003044' 303 02 0 00 000017'
1413 003045' 254 00 0 00 005630'
1414 003046' 200 01 0 00 000002'
1415 003047' 260 17 0 00 000000*
1416 003050' 332 00 0 00 000000*
1417 003051' 254 00 0 00 003064'
1418 003052' 554 01 0 00 000000#
1419 003053' 332 00 0 00 000001'
1420 003054' 334 01 0 00 006500'
1421 003055' 200 01 0 00 006501'
1422 003056' 550 02 0 00 000000#
1423 003057' 332 00 0 00 000002'
1424 003060' 334 02 0 00 006500'
1425 003061' 20C 02 0 00 006501'
1426
1427 003062' 260 17 0 00 003027*
1428 003064' 263 17 0 00 000000
1429
1430
1431
1432 003065' 260 17 0 00 004435'
1433 003066' 260 17 0 00 000000*
1434 003067' 263 17 0 00 000000
1435
1436
1437
1438 003070' 260 17 0 00 004435'
1439 003071' 260 17 0 00 000000*
1440 003072' 263 17 0 00 000000

```

```

1441          SUBTTL TAKE Command
1442
1443          ;Here to process TAKE command
1444
1445          003073' 332 00 0 00 000372'      .TAKE: SKIPE      TAKIFN      ;In a TAKE already?
1446          003074' 254 00 0 00 006554'      CMDERR      <Nested TAKE files are illegal>
1447          003075' 402 00 0 00 001212'      SETZM      GJFBLK      ;Clear first word
1448          003076' 200 01 0 00 005565'      MOVE      S1,[GJFBLK+1] ;Set up to clear block
1449          003077' 251 01 0 00 001227'      BLT      S1,GJFBLK+GJFSIZ-1 ;Clear the block
1450          003100' 201 02 0 00 006563'      MOVEI      S2,[FLDDB. (.CMNOI,..<Point 7,[ASCIZ/commands from/]>)]
1451          003101' 260 17 0 00 004443'      $CALL      CMDPRS      ;Do the noise
1452          003102' 336 00 0 00 000365'      SKIPN      MONTYP      ;Skip if TOPS-20
1453          003103' 254 00 0 00 003133'      JRST      TAKE1      ;TOPS-10
1454
1455          ;TOPS-20 take command
1456
1457          003104' 205 01 0 00 100000      MOVX      S1,GJ%OLD      ;File must exist
1458          003105' 202 01 0 00 001212'      MOVEM      S1,GJFBLK+.GJGEN ; into flags word
1459          003106' 200 01 0 00 000401'      MOVE      S1,CSB+.CMIOJ ;Load I/O JFNs
1460          003107' 202 01 0 00 001213'      MOVEM      S1,GJFBLK+.GJSRC ; into block
1461          003110' 561 01 0 00 005566'      HRROI      S1,[ASCIZ/DFCIB/] ;Point at default file name
1462          003111' 202 01 0 00 001216'      MOVEM      S1,GJFBLK+.GJNAM ;Save for GTJFN
1463          003112' 561 01 0 00 006565'      HRROI      S1,[ASCIZ/CMD/] ;Default extension
1464          003113' 202 01 0 00 001217'      MOVEM      S1,GJFBLK+.GJEXT ;Save in GTJFN block
1465          003114' 561 01 0 00 005571'      HRROI      S1,[ASCIZ/DSK/] ;Get the default structure
1466          003115' 202 01 0 00 001214'      MOVEM      S1,GJFBLK+.GJDEV ;Save the device
1467          003116' 201 02 0 00 006570'      MOVEI      S2,[FLDDB. (.CMFIL,...<DFCIB.CMD>)] ;Input file type
1468          003117' 260 17 0 00 004443'      $CALL      CMDPRS      ;Hello GLXLIB
1469          003120' 260 17 0 00 004435'      $CALL      CMDCFM      ;Confirm that
1470          003121' 561 01 0 00 000030'      HRROI      S1,TAKFD+.FDSTG ;Point to the FD
1471          003122' 202 02 0 00 000372'      MOVEM      S2,TAKIFN      ;Save the take JFN
1472          003123' 400 03 0 00 000000'      SETZ      T1. ;Default format
1473          003124' 104 00 0 00 000030'      JFNS%      ;JFN to string
1474          003125' 320 16 0 00 003126'      ERJMP      .+1 ;Error, ignore it
1475          003126' 200 01 0 00 000372'      MOVE      S1,TAKIFN ;Get the JFN again
1476          003127' 200 02 0 00 006574'      MOVX      S2,OF%RD!FLD(7,OF%BSZ) ;Read 7 bit bytes
1477          003130' 104 00 0 00 000021'      OPENF%      ;Pry it open
1478          003131' 320 16 0 00 003156'      ERJMP      TAKE6 ;Error, punt
1479          003132' 254 00 0 00 003153'      JRST      TAKE3 ;Give startip message
  
```

1480
 1481
 1482
 1483 003133' 200 01 0 00 005601'
 1484 003134' 202 01 0 00 001214'
 1485 003135' 205 01 0 00 435544'
 1486 003136' 202 01 0 00 001215'
 1487 003137' 205 01 0 00 446353'
 1488 003140' 202 01 0 00 001213'
 1489 003141' 201 02 0 00 006575'
 1490 003142' 260 17 0 00 004443'
 1491 003143' 200 01 0 00 006601'
 1492 003144' 251 01 0 00 000044'
 1493 003145' 260 17 0 00 004435'
 1494 003146' 201 01 0 00 000005'
 1495 003147' 201 02 0 00 000022'
 1496 003150' 260 17 0 00 002470*
 1497 003151' 322 00 0 00 003156'
 1498 003152' 202 01 0 00 000372'
 1499
 1500
 1501
 1502 003153' 260 17 0 00 003062*
 1503 003155' 263 17 0 00 000000
 1504
 1505
 1506
 1507 003156' 260 17 0 00 003153*
 1508 003160' 254 00 0 00 003161'

;Here for TOPS-10 TAKE command

TAKE1: MOVE S1,[SIXBIT/DFCIB/] ;Get file name
 STORE S1,GJFBLK+.FDNAM ;Save in default block
 MOVS1 S1,'CMD' ;Get default extension
 STORE S1,GJFBLK+.FDEXT ;Save in block
 MOVS1 S1,'DSK' ;Get structure name
 STORE S1,GJFBLK+.FDSTR ;Save the structure
 MOVEI S2,[FLDDB. (.CMIFI,...,<DFCIB.CMD>)] ;Input file
 \$CALL CMDPRS ;Hello GLXLIB
 MOVE S1,[GJFBLK+,TAKFD] ;Set up to copy into FD
 BLT S1,TAKFD+GJFSIZ-1 ;Clear the block
 \$CALL CMDCFM ;Confirm the command
 MOVEI S1,FOB.SZ ;Load size of FOB
 MOVEI S2,TAKFOB ;Point to the FOB
 \$CALL F%IOPN ;Open up the thing
 JUMPF TAKE6 ;Punt if an error
 MOVEM S1,TAKIFN ;Save the IFN

;Here to give startup message and then return for commands.

TAKE3: \$TEXT (,<[Processing ^F/TAKFD/]>)
 \$RET ;Return to get commands from file

;Here if an error opening the file, etc.

TAKE6: \$TEXT (,<? ^E/S1/>)
 JRST TAKABT ;Give error message
 ;Abort the take

```

1509
1510 ;Here to abort a TAKE file, returns to TOPLVL
1511
1512 003161' 336 01 0 00 000372' TAKABT: SKIPN S1,TAKIFN ;Skip if an IFN there
1513 003162' 254 00 0 00 002020' JRST TOPLVL ;None there, restart now
1514 003163' 332 00 0 00 000365' SKIPE MONTYP ;Skip if TOPS-10
1515 003164' 104 00 0 00 000022' CLOSFX ;TOPS-20, close the file
1516 003165' 320 16 0 00 003166' ERJMP .+1 ;Ignore errors
1517 003166' 336 00 0 00 000365' SKIPN MONTYP ;Skip if TOPS-20
1518 003167' 260 17 0 00 002401* $CALL FXREL ;TOPS-10, release the file
1519 003170' 402 00 0 00 000372' SETZM TAKIFN ;No longer an IFN/JFN
1520 003171' 254 00 0 00 002020' JRST TOPLVL ;Restart the parsing
1521
1522 ;Here to check if we had an EOF on the take file.
1523
1524 003172' 336 00 0 00 000365' TAKEOF: SKIPN MONTYP ;Skip if TOPS-20
1525 003173' 254 00 0 00 003202' JRST TAKE01 ;TOPS-10
1526 003174' 336 01 0 00 000372' SKIPN S1,TAKIFN ;Get input file JFN for take file
1527 003175' 324740 000000* $RETF ;No take file, so not EOF
1528 003176' 104 00 0 00 000024 GTSTS% ;Get the file's status
1529 003177' 607 02 0 00 001000 TXNN S2,GS%EOF ;At end of file?
1530 003200' 324740 003175* $RETF ;Not at end of file, return false
1531 003201' 324740 002473* $RETT ;At end of file, return true
1532
1533 003202' 332 00 0 00 000372' TAKE01: SKIPE TAKIFN ;Skip if no take file
1534 003203' 302 01 0 00 000C01 CAIE S1,EREOF$ ;End of file?
1535 003204' 324740 003200* $RETF ;No take file or not EOF
1536 003205' 324740 003201* $RETT ;Yes, take file and EOF

```



```

1537          SUBTTL  Quit Command
1538
1539          ;Here for ^C interrupts for TOPS-10
1540
1541 003206' 332 00 0 00 000252' CCIN10: SKIPE  FIVECT+CCOFFS+.PSVIS ;Was the ^C typed at a prompt?
1542 003207' 051 03 0 00 006624'      OUTSTR  [ASCIZ/^C/] ;Yes, tell him a control C
1543 003210' 260 17 0 00 002652'      $CALL  UNLKIT ;Unlock the program
1544 003211' 254 00 0 00 003215'      JRST   QUIT1 ;Exit the program
1545
1546          ;Here for quit command.
1547
1548 003212' 201 02 0 00 006630' .QUIT: MOVEI  S2,[FLDDB. (.CMNOI,,<Point 7,[ASCIZ/to monitor/]>)]
1549 003213' 260 17 0 00 004443'      $CALL  CMDPRS ;Output the noise words
1550 003214' 260 17 0 00 004435'      $CALL  CMDCFM ;Confirm the diagnostic exit
1551          ;:***: $CALL  SPR4 ;Go make a spear entry for this
1552 003215' 260 17 0 00 004402' QUIT1: $CALL  LOGCLS ;Disable logging
1553 003216' 254 00 0 00 000000*      JRST   I%EXIT ;Exit program

```

```

1554          SUBTTL  Send Command
1555
1556          ;SEND Command
1557
1558          003217' 201 02 0 00 006633' .SEND:  GUIDE  <type>          ;noise word
1559          003220' 260 17 0 00 004443'
1560          003221' 201 02 0 00 006635'      MOVEI  S2,[FLDDB.(.CMKEY,,SELSEN)] ;Parse table
1561          003222' 260 17 0 00 004443'      $CALL  CMDPRS          ;Parse that command
1562          003223' 550 01 0 02 000000      HRRZ   S1,(S2)          ;Set up to switch or retry routine
1563          003224' 254 00 0 01 000000      JRST   (S1)          ;Go do the routine
1564
1565          ;Here is the table that for the send command
1566
1567          003225' 000003' 000003' SELSEN:  $STAB          ;Start of the send table
1568          003226' 006637' 003237'      KEYTAB  .DATA,DATA          ;Routine to send data
1569          003227' 006640' 003234'      KEYTAB  .SDATA,DATAGRAM      ;Routine to send datagram
1570          003230' 006642' 003231'      KEYTAB  .SMESS,MESSAGE      ;Routine to send message
1571          000004'      $ETAB          ;End of send table
1572
1573          ;Routine to send message
1574
1575          003231' 260 17 0 00 004435' .SMESS:  $CALL  CMDCFM          ;Command confirmation
1576          003232' 260 17 0 00 000000*  $CALL  SNDMSG          ;Go send the message
1577          003233' 263 17 0 00 000000      $RET
1578
1579          ;Routine to send datagram
1580
1581          003234' 260 17 0 00 004435' .SDATA:  $CALL  CMDCFM          ;Command confirmation
1582          003235' 260 17 0 00 000000*  $CALL  SNDDAT          ;Go send the datagram
1583          003236' 263 17 0 00 000000      $RET

```

```
1584
1585 ;SEND DATA command
1586
1587 003237' 201 02 0 00 006645' .DATA: GUIDE <to> ;Noise word
1588 003240' 260 17 0 00 004443'
1589 003241' 201 02 0 00 006647' MOVEI S2,[FLDDB.(.CMKEY,,DATSEN)] ;parse table
1590 003242' 260 17 0 00 004443' $CALL CMDPRS ;Parse the field please
1591 003243' 550 01 0 02 000000 HRRZ S1,(S2) ;Set up to switch or retry routine
1592 003244' 260 17 0 00 004435' $CALL CMDCFM ;Confirm that
1593 003245' 254 00 0 01 000000 JRST (S1) ;Go do the routine
1594
1595 ;Here is the table that for the send command
1596
1597 003246' 000002 000002 DATSEN: $STAB ;Start of the send table
1598 003247' 006651' 000000* KEYTAB DATSE,EXERCISER-NODE ;Routine to send data
1599 003250' 006654' 000000* KEYTAB DATSR,RESPONDER-NODE ;Routine to send data
1600 000003 $ETAB ;End of send table
```

```
1601          SUBTTL  Connect/Disconnect Commands
1602
1603          ;Routine to connect to node selected by select node command
1604
1605 003251' 260 17 0 00 004435' .CONN: $CALL  CMDCFM          ;Command confirmation
1606 003252' 260 17 0 00 000000* $CALL  DOCON          ;DFCIB1.MAC
1607 003253' 263 17 0 00 000000 $RET
1608
1609          ;Routine to disconnect to node
1610
1611 003254' 260 17 0 00 004435' .DCONN: $CALL  CMDCFM          ;Command confirmation
1612 003255' 260 17 0 00 000000* $CALL  DISC          ;DFCIB1.MAC
1613 003256' 263 17 0 00 000000 $RET
```

```

1614          SUBTTL Wait Command
1615
1616          ;Here for 'WAIT'
1617
1618          003257' 201 02 0 00 006660' .WAIT: GUIDE (for) ;Noise
1619          003260' 260 17 0 00 004443'
1620          003261' 201 02 0 00 006662'          MOVEI S2,[FLDDB.(.CMKEY,,SELWAI)] ;parse table
1621          003262' 260 17 0 00 004443'          $CALL CMDPRS ;Parse that please
1622          003263' 550 01 0 02 000000          HRRZ S1,(S2) ;Set up to switch or retry routine
1623          003264' 254 00 0 01 000000          JRST (S1) ;Go do the routine
1624
1625          ;Here is the table for the wait command
1626
1627          003265' 000002 000002 SELWAI: $STAB ;Start of the wait table
1628          003266' 006664' 003270' KEYTAB .WCS,CONNECTION-STATUS ;Routine for connection status
1629          003267' 006670' 003340' KEYTAB .WES,EVENT-STATUS ;Routine for event status
1630          000003 SETAB ;End of wait table
1631
1632          ;Here for 'WAIT CONNECTION-STATUS'
1633
1634          003270' 201 02 0 00 006673' .WCS: MOVEI S2 [FLDDB.(.CMKEY,..WSC10)] ;Parse table
1635          003271' 260 17 0 00 004443' $CALL CMDPRS ;Parse that please
1636          003272' 550 01 0 02 000000 HRRZ S1,(S2) ;Set up to switch or retry routine
1637          003273' 254 00 0 01 000000 JRST (S1) ;Go do the routine
1638
1639          ;Here is the table for the status info on connection command
1640
1641          003274' 000002 000002 .WSC10: $STAB ;Start of the table
1642          003275' 006675' 003277' KEYTAB .WCS1,FLAG ;Routine for flag status
1643          003276' 006676' 003314' KEYTAB .WCS2,STATE ;Routine for state status
1644          000003 SETAB ;End of table
1645
1646          ;Here for 'WAIT CONNECTION-STATUS FLAG'
1647
1648          003277' 201 02 0 00 006677' .WCS1: MOVEI S2,[FLDDB.(.CMKEY,,WSC12)] ;Parse table
1649          003300' 260 17 0 00 004443' $CALL CMDPRS ;Parse the field please
1650          003301' 514 01 0 02 000000 HRRZ S1,(S2) ;Get bit
1651          003302' 202 01 0 00 003306' MOVEM S1,CHKST ;Save it
1652          003303' 260 17 0 00 004435' $CALL CMDCFM ;Command confirmation
1653          003304' 260 17 0 00 000000* $CALL CWAIT ;Wait for it
1654          003305' 263 17 0 00 000000 $RET
1655
1656          003306' 000 00 0 00 000000 CHKST: Z ;The status to check. If bits 0-3
1657          ; are set, check status flag. If not
1658          ; check state
1659
1660          003307' 000004 000004 WSC12: $STAB ;Begining of status name table
1661          003310' 006701' 200000 KEYTAB 1B19,DATAGRAM-AVAILABLE-FLAG
1662          003311' 006706' 100000 KEYTAB 1B20,DMA-TRANSFER-COMPLETE
1663          003312' 006713' 040000 KEYTAB 1B21,EVENT-PENDING-FLAG
1664          003313' 006717' 400000 KEYTAB 1B18,MESSAGE-AVAILABLE-FLAG
1665          000005 SETAB ;End of status name table
  
```



```

1666
1667
1668
1669 003314' 201 02 0 00 006724'
1670 003315' 260 17 0 00 004443'
1671 003316' 550 01 0 02 000000
1672 003317' 202 01 0 00 003306'
1673 003320' 260 17 0 00 004435'
1674 003321' 260 17 0 00 003304*
1675 003322' 263 17 0 00 000000
1676
1677 003323' 000014 000014
1678 003324' 006726' 000006
1679 003325' 006732' 000001
1680 003326' 006734' 000004
1681 003327' 006741' 000003
1682 003330' 006746' 000005
1683 003331' 006754' 000010
1684 003332' 006760' 000012
1685 003333' 006766' 000011
1686 003334' 006773' 000013
1687 003335' 007001' 000002
1688 003336' 007003' 000007
1689 003337' 007007' 000014
1690
1691
1692
1693
1694 003340' 201 02 0 00 007016'
1695 003341' 260 17 0 00 004443'
1696 003342' 550 01 0 02 000000
1697 003343' 202 01 0 00 000363'
1698 003344' 260 17 0 00 004435'
1699 003345' 260 17 0 00 000000*
1700 003346' 263 17 0 00 000000
1701
1702 003347' 000015 000015
1703 003350' 007020' 000002
1704 003351' 007024' 000003
1705 003352' 007031' 000004
1706 003353' 007036' 000014
1707 003354' 007042' 000006
1708 003355' 007046' 000015
1709 003356' 007053' 000005
1710 003357' 007062' 000010
1711 003360' 007066' 000007
1712 003361' 007072' 000011
1713 003362' 007076' 000013
1714 003363' 007103' 000012
1715 003364' 007111' 000001
1716
  
```

;Here for 'WAIT CONNECTION-STATUS STATE'

```

.WCS2: MOVEI S2,[FLDDB.(.CMKEY,,WSC22)] ;Parse table
        $CALL CMDPRS ;Parse the field please
        HRRZ S1,(S2) ;Get bit
        MOVEM S1,CHKST ;Save it
        $CALL CMDCFM ;Command confirmation
        $CALL CWAIT ;Wait for it
        $RET
  
```

```

WSC22: $STAB ;Begining of flags table
        KEYTAB 6,ACCEPT-REQUEST-SENT
        KEYTAB 1,CLOSED
        KEYTAB 4,CONNECT-REQUEST-RECEIVED
        KEYTAB 3,CONNECT-REQUEST-SENT
        KEYTAB 5,CONNECT-RESPONSE-RECEIVED
        KEYTAB 10,CONNECTION-IS-OPEN
        KEYTAB 12,DISCONNECT-REQUEST-RECEIVED
        KEYTAB 11,DISCONNECT-REQUEST-SENT
        KEYTAB 13,DISCONNECT-RESPONSE-RECEIVED
        KEYTAB 2,LISTENING
        KEYTAB 7,REJECT-REQUEST-SENT
        KEYTAB 14,WAITING-FOR-DISCONNECT-RESPONSE
        $ETAB ;End of flags name table
  
```

; 'WAIT-FOR EVENT'

```

.WES: MOVEI S2,[FLDDB.(.CMKEY,,WEC12)] ;Parse table
        $CALL CMDPRS ;Parse the field please
        HRRZ S1,(S2) ;Get bit
        MOVEM S1,CHKEV ;Save it
        $CALL CMDCFM ;Command confirmation
        $CALL EWAIT ;Do the work
        $RET
  
```

```

WEC12: $STAB ;Begining of event name table
        KEYTAB .SECTL,CONNECT-TO-LISTEN
        KEYTAB .SECRA,CONNECTION-WAS-ACCEPTED
        KEYTAB .SECR,CONNECTION-WAS-REJECTED
        KEYTAB .SECIA,CREDIT-IS-AVAILABLE
        KEYTAB .SELCL,LITTLE-CREDIT-LEFT
        KEYTAB .SEMD,MAINT-DATA-XFER-COMplete
        KEYTAB .SEMSC,MESSAGE-DATAGRAM-SEND-COMplete
        KEYTAB .SENC,NOde-CAME-ONLINE
        KEYTAB .SENWO,NOde-WENT-OFFLINE
        KEYTAB .SEOSD,OK-TO-SEND-DATA
        KEYTAB .SEPBC,PORT-BROKE-CONNECTION
        KEYTAB .SERID,REMOTE-INITIATED-DISCONNECT
        KEYTAB .SEVCC,VC-BROKEN
        $ETAB ;End of event name table
  
```

```

1717
1718 ;Build CTP packet
1719
1720 003365' 201 02 0 00 007117' .BUILD: GUIDE <CTP packet type> ;Noise word
1721 003366' 260 17 0 00 004443'
1722 003367' 201 02 0 00 007125' MOVEI S2,[FLDDB.(.CMNUM.,^D10,<Packet type, 0-13>)]
1723 003370' 260 17 0 00 004443' $CALL CMDPRS ;Parse the field please
1724 003371' 303 02 0 00 000015' CAILE S2,^D13 ;Skip if 13 or less
1725 003372' 254 00 0 00 007143' CMDERR <Packet type out of range 0-13> ;Out of range
1726 003373' 202 02 0 00 000000* MOVEM S2,OPCODE ;Save it in CIEX1.MAC
1727 003374' 260 17 1 02 007147' $CALL @[EXP BU0,BU1,BU2,BU3,BU4,BU5,BU6,BU7,BU8,BU9,BU10,BU11,BU12,BU13](S2)
1728 003375' 260 17 0 00 000000* $CALL BLDCTP ;CIEX1.MAC
1729 003376' 263 17 0 00 000000 $RET
1730
1731 ;Build CTP packet type #0, #12, #13
1732
1733 003377' 254 00 0 00 007201' BU12: BU13: BU0: CMDERR <That packet type is not implemented>
1734
1735 ;Build CTP packet type #1
1736
1737 003400' 304 00 0 00 000000 BU1: $SAVE <T1,S2,S1> ;Save ACs
1738 003406' 260 17 0 00 003563' $CALL BTYPE ;Buffer type
1739 003407' 260 17 0 00 003662' $CALL BGENF ;Generate function
1740 003410' 260 17 0 00 003673' $CALL BGENC ;Generate constant
1741 003411' 260 17 0 00 003704' $CALL BBGENL ;Generate buffer length
1742 003412' 260 17 0 00 004435' $CALL CMDCFM ;We expect command confirmation here
1743 003413' 263 17 0 00 000000 $RET
1744
1745 ;Build CTP packet type #2
1746
1747 003414' 304 00 0 00 000000 BU2: $SAVE <T1,S2,S1> ;Save ACs
1748 003422' 260 17 0 00 003563' $CALL BTYPE ;Buffer type
1749 003423' 260 17 0 00 003704' $CALL BBGENL ;Generate buffer length
1750 003424' 260 17 0 00 004435' $CALL CMDCFM ;We expect command confirmation here
1751 003425' 263 17 0 00 000000 $RET

```

```
1752
1753      ;Build CTP packet type #3
1754
1755      BU3:  $SAVE  <T1,S2,S1>      ;Save ACs
1756            $CALL  BDELAY           ;Delay count
1757            $CALL  BRCNT            ;Repeat count
1758            $CALL  BMOVT3          ;Buffer move type
1759            $CALL  BPKTS            ;Base packet size
1760            $CALL  BOTHER           ;Other node
1761            $CALL  BPKTM            ;Packet size multiple
1762            $CALL  BBGENL           ;Generate buffer length
1763            $CALL  BLOFFS           ;Local buffer offset
1764            $CALL  BROFFS           ;Remote buffer offset
1765            $CALL  CMDCFM           ;We expect command confirmation here
1766            $RET
1767
1768
1769      ;Build CTP packet type #4 and #5
1770
1771      BU5:
1772      BU4:  $SAVE  <T1,S2,S1>      ;Save ACs
1773            $CALL  RDELAY           ;Delay count
1774            $CALL  BRCNT            ;Repeat count
1775            $CALL  BGENF            ;Gen function
1776            $CALL  BGENC            ;Gen constant
1777            $CALL  BMLN             ;Message length
1778            $CALL  BIDATA           ;Image data
1779            $CALL  CMDCFM           ;We expect command confirmation here
1780            $RET
1781
1782      ;Build CTP packet type #6
1783
1784      BU6:  $SAVE  <T1,S2,S1>      ;Save ACs
1785            $CALL  BDELAY           ;Delay count
1786            $CALL  BRCNT            ;Repeat count
1787            $CALL  BEXTEN           ;Reset type
1788            $CALL  BOTHER           ;Other node
1789            $CALL  CMDCFM           ;We expect command confirmation here
1790            $RET
1791
1755 003426' 304 00 0 00 000000
1756 003434' 260 17 0 00 003737'
1757 003435' 260 17 0 00 003750'
1758 003436' 260 17 0 00 003616'
1759 003437' 260 17 0 00 003627'
1760 003440' 260 17 0 00 003651'
1761 003441' 260 17 0 00 003640'
1762 003442' 260 17 0 00 003704'
1763 003443' 260 17 0 00 003761'
1764 003444' 260 17 0 00 003772'
1765 003445' 260 17 0 00 004435'
1766 003446' 263 17 0 00 000000
1771 003447'
1772 003447' 304 00 0 00 000000
1773 003455' 260 17 0 00 003737'
1774 003456' 260 17 0 00 003750'
1775 003457' 260 17 0 00 003662'
1776 003460' 260 17 0 00 003673'
1777 003461' 260 17 0 00 003715'
1778 003462' 260 17 0 00 004003'
1779 003463' 260 17 0 00 004435'
1780 003464' 263 17 0 00 000000
1784 003465' 304 00 0 00 000000
1785 003473' 260 17 0 00 003737'
1786 003474' 260 17 0 00 003750'
1787 003475' 260 17 0 00 003605'
1788 003476' 260 17 0 00 003651'
1789 003477' 260 17 0 00 004435'
1790 003500' 263 17 0 00 000000
```

```
1791
1792 ;Build CTP packet type #7
1793
1794 003501' 304 00 0 00 000000 BU7: $SAVE <T1,S2,S1> ;Save ACs
1795 003507' 260 17 0 00 003737' $CALL BDELAY ;Delay count
1796 003510' 260 17 0 00 003750' $CALL BRCNT ;Repeat count
1797 003511' 260 17 0 00 003605' $CALL BEXTEN ;Resettype
1798 003512' 260 17 0 00 003651' $CALL BOTHER ;Other node
1799 003513' 332 00 0 00 000000* $CALL BOTHER ;Skip if node default address to use
1800 003514' 260 17 0 00 003726' $CALL BSTADR ;Start addr
1801 003515' 260 17 0 00 004435' $CALL CMDCFM ;We expect command confirmation here
1802 003516' 263 17 0 00 000000 $RET
1803
1804 ;Build CTP packet type #8
1805
1806 003517' 304 00 0 00 000000 BU8: $SAVE <T1,S2,S1> ;Save ACs
1807 003525' 260 17 0 00 003574' $CALL BACTF ;Activity type
1808 003526' 260 17 0 00 004435' $CALL CMDCFM ;We expect command confirmation here
1809 003527' 263 17 0 00 000000 $RET
1810
1811 ;Build CTP packet type #9
1812
1813 003530' 304 00 0 00 000000 BU9: $SAVE <T1,S2,S1> ;Save ACs
1814 003536' 260 17 0 00 003651' $CALL BOTHER ;Other node
1815 003537' 260 17 0 00 004435' $CALL CMDCFM ;We expect command confirmation here
1816 003540' 263 17 0 00 000000 $RET
1817
1818 ;Build CTP packet type #10
1819
1820 003541' 304 00 0 00 000000 BU10: $SAVE <T1,S2,S1> ;Save ACs
1821 003547' 260 17 0 00 003651' $CALL BOTHER ;Other node
1822 003550' 260 17 0 00 004435' $CALL CMDCFM ;We expect command confirmation here
1823 003551' 263 17 0 00 000000 $RET
1824
1825 ;Build CTP packet type #11
1826
1827 003552' 304 00 0 00 000000 BU11: $SAVE <T1,S2,S1> ;Save ACs
1828 003560' 260 17 0 00 003651' $CALL BOTHER ;Other node
1829 003561' 260 17 0 00 004435' $CALL CMDCFM ;We expect command confirmation here
1830 003562' 263 17 0 00 000000 $RET
```



```

1831
1832 003563' 201 02 0 00 007220'
1833 003564' 260 17 0 00 004443'
1834 003565' 201 02 0 00 007226'
1835 003566' 260 17 0 00 004443'
1836 003567' 301 02 0 00 000001
1837 003570' 303 02 0 00 000003
1838 003571' 254 00 0 00 007244'
1839 003572' 202 02 0 00 000000*
1840 003573' 263 17 0 00 000000
1841
1842 003574' 201 02 0 00 007254'
1843 003575' 260 17 0 00 004443'
1844 003576' 201 02 0 00 007263'
1845 003577' 260 17 0 00 004443'
1846 003600' 301 02 0 00 000000
1847 003601' 303 02 0 00 000001
1848 003602' 254 00 0 00 007301'
1849 003603' 202 02 0 00 000000*
1850 003604' 263 17 0 00 000000
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863 003605' 201 02 0 00 007310'
1864 003606' 260 17 0 00 004443'
1865 003607' 201 02 0 00 007316'
1866 003610' 260 17 0 00 004443'
1867 003611' 301 02 0 00 000000
1868 003612' 303 02 0 00 000001
1869 003613' 254 00 0 00 007332'
1870 003614' 202 02 0 00 003513*
1871 003615' 263 17 0 00 000000
1872
1873 003616' 201 02 0 00 007343'
1874 003617' 260 17 0 00 004443'
1875 003620' 201 02 0 00 007352'
1876 003621' 260 17 0 00 004443'
1877 003622' 301 02 0 00 000000
1878 003623' 303 02 0 00 000003
1879 003624' 254 00 0 00 007371'
1880 003625' 202 02 0 00 000000*
1881 003626' 263 17 0 00 000000
  
```

```

BTYPE: GUIDE <Buffer type 1-3> ;Noise word
        MOVEI S2,[FLDDB.(.CMNUM,,^D10,<Buffer type, 1-3>)]
        $CALL CMDPRS ;Parse the field please
        CAIL S2,^D1 ;Skip if lt 1
        CAILE S2,^D3 ;Skip if 3 or less
        CMDERR <Buffer type out of range 1-3> ;out of range
        MOVEM S2,BUFTYP ;Save it in CIEX1.MAC
        $RET

BACTF: GUIDE <Activity type 0-1> ;Noise word
        MOVEI S2,[FLDDB.(.CMNUM,CM%SDH,^D10,<Activity type 0 or 1>)]
        $CALL CMDPRS ;Parse the field please
        CAIL S2,^D0 ;Skip if lt 0
        CAILE S2,^D1 ;Skip if 1 or less
        CMDERR <Activity type out of range 0-1> ;out of range
        MOVEM S2,ACTFLG ;Save it in CIEX1.MAC
        $RET

repeat 0,<
BMOVET: GUIDE <Buffer move type 0-1> ;Noise word
        MOVEI S2,[FLDDB.(.CMNUM,CM%SDH,^D10,<Buffer move type 0 or 1>)]
        $CALL CMDPRS ;Parse the field please
        CAIL S2,^D0 ;Skip if lt 0
        CAILE S2,^D1 ;Skip if 1 or less
        CMDERR <Buffer move type not 0 or 1> ;out of range
        MOVEM S2,MOVTP ;Save it in CIEX1.MAC
        $RET
>;end of repeat 0

BEXTEN: GUIDE <Reset type 0-1> ;Noise word
        MOVEI S2,[FLDDB.(.CMNUM,CM%SDH,^D10,<Reset type 0 or 1>)]
        $CALL CMDPRS ;Parse the field please
        CAIL S2,^D0 ;Skip if lt 0
        CAILE S2,^D1 ;Skip if 1 or less
        CMDERR <Reset type not 0 or 1> ;out of range
        MOVEM S2,EXTEND ;Save it in CIEX1.MAC
        $RET

BMOVT3: GUIDE <Buffer move type 0-3> ;Noise word
        MOVEI S2,[FLDDB.(.CMNUM,,^D10,<Buffer move type, 0-3>)]
        $CALL CMDPRS ;Parse the field please
        CAIL S2,^D0 ;Skip if lt 0
        CAILE S2,^D3 ;Skip if 3 or less
        CMDERR <Buffer move type not in range 0-3> ;out of range
        MOVEM S2,MOVTP ;Save it in CIEX1.MAC
        $RET
  
```


1882							
1883	003627'	201	02	0	00	007402'	
1884	003630'	260	17	0	00	004443'	
1885	003631'	201	02	0	00	007411'	
1886	003632'	260	17	0	00	004443'	
1887	003633'	301	02	0	00	000000	
1888	003634'	303	02	0	00	000001	
1889	003635'	254	00	0	00	007426'	
1890	003636'	202	02	0	00	000000*	
1891	003637'	263	17	0	00	000000	
1892							
1893	003640'	201	02	0	00	007440'	
1894	003641'	260	17	0	00	004443'	
1895	003642'	201	02	0	00	007450'	
1896	003643'	260	17	0	00	004443'	
1897	003644'	301	02	0	00	000001	
1898	003645'	303	02	0	00	000377	
1899	003646'	254	00	0	00	007470'	
1900	003647'	202	02	0	00	000000*	
1901	003650'	263	17	0	00	000000	
1902							
1903	003651'	201	02	0	00	007500'	
1904	003652'	260	17	0	00	004443'	
1905	003653'	201	02	0	00	007506'	
1906	003654'	260	17	0	00	004443'	
1907	003655'	301	02	0	00	000000	
1908	003656'	303	02	0	00	000017	
1909	003657'	254	00	0	00	005630'	
1910	003660'	202	02	0	00	000000*	
1911	003661'	263	17	0	00	000000	
1912							
1913	003662'	201	02	0	00	007516'	
1914	003663'	260	17	0	00	004443'	
1915	003664'	201	02	0	00	007525'	
1916	003665'	260	17	0	00	004443'	
1917	003666'	301	02	0	00	000000	
1918	003667'	303	02	0	00	000003	
1919	003670'	254	00	0	00	007544'	
1920	003671'	202	02	0	00	000000*	
1921	003672'	263	17	0	00	000000	
1922							
1923	003673'	201	02	0	00	007555'	
1924	003674'	260	17	0	00	004443'	
1925	003675'	260	17	0	00	004443'	
1926	003676'	201	02	0	00	007564'	
1927	003677'	301	02	0	00	000000	
1928	003700'	303	02	0	00	000377	
1929	003701'	254	00	0	00	007603'	
1930	003702'	202	02	0	00	000000*	
1931	003703'	263	17	0	00	000000	

```

BPKTS:  GUIDE    <Base packet size 0-1> ;Noise word

        MOVEI    S2,[FLDDB.(.CMNUM,CM%SDH,^D10,<Base packet size 0 or 1>)]
        $CALL    CMDPRS                      ;Parse the field please
        CAIL     S2,^D0                      ;Skip if lt 0
        CAILE    S2,^D1                      ;Skip if 1 or less
        CMDERR   <Base packet size not 0 or 1> ;out of range
        MOVEM    S2,PKTSIZ                   ;Save it in CIEX1.MAC
        $RET

BPKTM:  GUIDE    <Packet size multiple 1-255> ;Noise word

        MOVEI    S2,[FLDDB.(.CMNUM,,^D10,<Packet size multiple, 1-255>)]
        $CALL    CMDPRS                      ;Parse the field please
        CAIL     S2,^D1                      ;Skip if lt 1
        CAILE    S2,^D255                    ;Skip if 255 or less
        CMDERR   <Packet size out of range 1-255 decimal> ;out of range
        MOVEM    S2,PKTMLT                   ;Save it in CIEX1.MAC
        $RET

BOTHER: GUIDE    <Other node 0-15>           ;Noise word

        MOVEI    S2,[FLDDB.(.CMNUM,,^D10,<Other node, 0-15>)]
        $CALL    CMDPRS                      ;Parse the field please
        CAIL     S2,^D0                      ;Skip if less than 0
        CAILE    S2,^D15                    ;Skip if 15 or less
        CMDERR   <Node number out of range 0-15> ;out of range
        MOVEM    S2,OTHNOD                   ;Save it in CIEX1.MAC
        $RET

BGENF:  GUIDE    <Generate Function 0-3> ;Noise word

        MOVEI    S2,[FLDDB.(.CMNUM,,^D10,<Generate Function, 0-3>)]
        $CALL    CMDPRS                      ;Parse the field please
        CAIL     S2,^D0                      ;Skip if lt 0
        CAILE    S2,^D3                      ;Skip if 3 or less
        CMDERR   <Generate function not in range 0-3> ;out of range
        MOVEM    S2,GENFUN                   ;Save it in CIEX1.MAC
        $RET

BGENC:  GUIDE    <Generate Constant 0-377> ;Noise word

        $CALL    CMDPRS                      ;Parse the field please
        MOVEI    S2,[FLDDB.(.CMNUM,,^D8,<Generate Constant, 0-377>)]
        CAIL     S2,^D0                      ;Skip if lt 0
        CAILE    S2,377                      ;Skip if 377 or less
        CMDERR   <Generate constant not in range 0-377> ;out of range
        MOVEM    S2,GNCNST                   ;Save it in CIEX1.MAC
        $RET

```

Year	Address	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418
------	---------	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------


```
2011          SUBTTL  Read Command
2012
2013          ;Read datagram or message
2014
2015          .READ:  GUIDE    <t pe>                ;Noise word
2016          MOVEI    S2,[FLDDB.(.CMKEY,,SELREA)] ;Parse table
2017          $CALL    CMDPRS                ;Parse the field please
2018          HRRZ     S1,(S2)                ;Set up for datagram or message routine
2019          $CALL    CMDCFM                ;Command confirmation
2020          JRST     (S1)                    ;Go do the routine
2021
2022          ;Here is the table that for the read command
2023
2024          SELREA:  $STAB                ;Start of the send table
2025          KEYTAB   .RDATA,DATAGRAM      ;Routine to read datagram
2026          KEYTAB   .RMESS,MESSAGE       ;Routine to read message
2027          $ETAB                ;End of send table
2028
2029          ;Routine to read message
2030
2031          .RMESS:  $CALL    READM                ;Go read the message CIEX1.MAC
2032          $CALL    REQUEM                ;Requeue the buffer
2033          $RET
2034
2035          ;Routine to read datagram
2036
2037          .RDATA:  $CALL    READD                ;Go read the datagram CIEX1.MAC
2038          $CALL    REQIED                ;Requeue the buffer
2039          $RET
2040
```

Address	Offset	Field	Value
004013	201	02	0 00 006633
004014	260	17	0 00 004443
004015	201	02	0 00 010207
004016	260	17	0 00 004443
004017	550	01	0 02 000000
004020	260	7	0 00 004435
004021	254	00	0 01 000000
004022	000002		000002
004023	006640		004030
004024	006342		004025
			000003
004025	260	17	0 00 000000*
004026	260	17	0 00 000000*
004027	263	17	0 00 000000
004030	260	17	0 00 000000*
004031	260	17	0 00 000000*
004032	263	17	0 00 000000


```

2041          SUBTTL Requeue Command
2042
2043          ;Requeue message or datagram buffer
2044
2045          004033' 201 02 0 00 010211' .REQ: MOVEI S2,[FLDDB.(.CMKEY,..REQ10)] ;Parse field
2046          004034' 260 17 0 00 004443' $CALL CMDPRS ;Parse the field please
2047          004035' 550 01 0 02 000000 HRRZ S1,(S2) ;Set up to switch or retry routine
2048          004036' 260 17 0 00 004435' $CALL CMDCFM ;Command confirmation
2049          004037' 254 00 0 01 000000 JRST (S1) ;Go do the routine
2050
2051          ;Here is the table for the REQUEUE command
2052
2053          004040' 000002 000002 .REQ10: $STAB ;Start of the table
2054          004041' 010213' 004043' KEYTAB .REQS1,DATAGRAM-BUFFER ;Routine for requeue datagram buffer
2055          004042' 010217' 004045' KEYTAB .REQS2,MESSAGE-BUFFER ;Routine for requeue message buffer
2056          000003 $ETAB ;End of table
2057
2058          ;Requeue datagram-buffer command
2059
2060          004043' 260 17 0 00 004031* .REQS1: $CALL REQUED ;CIEX1.MAC
2061          004044' 263 17 0 00 000000 $RET
2062
2063          ;Requeue message-buffer command
2064
2065          004045' 260 17 0 00 004026* .REQS2: $CALL REQUEM ;CIEX1.MAC
2066          004046' 263 17 0 00 000000 $RET
2067
2068          ;RRET Return-message command. !!! Responder only !!!
2069
2070          004047' 260 17 0 00 004435' .RRET: $CALL CMDCFM ;We expect command confirmation here
2071          004050' 263 17 0 00 000000* $CALL RESPR ;Go return the message
2072          004051' 263 17 0 00 000000 $PET ;Return to toplevel
  
```


2073 SUBTTL Type Command

2074
 2075 776000 000000 BYTE1==377B7
 2076 001774 000000 BYTE2==377B15
 2077 000003 770000 BYTE3==377B23
 2078 007760 BYTE4==377B31
 2079 000017 BYTE5==017B35

;Type out a buffer

2083	004052'	201	02	0	00	010225'	.TYPE:	GUIDE	<buffer name>	;Noise word
2084	004053'	260	17	0	00	004443'				
2085	004054'	201	02	0	00	010227'		MOVEI	S2,[FLDDB.(.CMKEY,,TYPE1)]	
2086	004055'	260	17	0	00	004443'		\$CALL	CMDPRS	;Parse the field please
2087	004056'	550	01	0	02	000000		HRRZ	S1,(S2)	;Set up the address of the buffer
2088	004057'	260	17	0	00	004435'		\$CALL	CMDCFM	;Command confirmation
2089	004060'	306	01	0	00	000364'		CAIN	S1,MAPBBA	;MAPBB Address
2090	004061'	200	01	0	00	000364'		MOVE	S1,MAPBBA	;Yes, get address
2091	004062'	306	01	0	00	000000*		CAIN	S1,RETBUF	;Is it RETBUF ?
2092	004063'	200	01	0	00	004062*		MOVE	S1,RETBUF	;Yes, then get the contents of it
2093	004064'	402	00	0	00	000002		SETZM	S2	;Zero index
2094	004065'	402	00	0	00	000003		SETZM	T1	;Zero byte count
2095	004066'						TYPE1:	\$TEXT	(,<^D2R /T1/^013R0/(S1)/^04R /(S1),BYTE1/^04R /(S1),BYTE2/^04R /(S1),BYTE3/^04R /(S1),BYTE4/^03R /(S1),BYTE5/^A>)	
2096	004066'	260	17	0	00	003156*		AOS	S1	;Increment address
2097	004070'	350	00	0	00	000001		AOS	S2	;Increment counter
2098	004071'	350	00	0	00	000002		ADDI	T1,4	;Increment byte count
2099	004072'	271	03	0	00	000004		\$TEXT	(,<^D2R /T1/^013R0/(S1)/^04R /(S1),BYTE1/^04R /(S1),BYTE2/^04R /(S1),BYTE3/^04R /(S1),BYTE4/^03R /(S1),BYTE5/>)	
2100										
2101	004073'	260	17	0	00	004066*		AOS	S1	;Increment address
2102	004075'	350	00	0	00	000001		AOS	S2	;Increment counter
2103	004076'	350	00	0	00	000002		ADDI	T1,4	;Increment byte count
2104	004077'	271	03	0	00	000004		CAMGE	S2,TYLEN	;Printed all lines ?
2105	004100'	315	02	0	00	002112'		JRST	TYPE1	; No
2106	004101'	254	00	0	00	004066'		\$RET		
2107	004102'	263	17	0	00	000000				

2108
 2109
 2110
 2111
 2112
 2113
 2114
 2115
 2116
 2117
 2118
 2119
 2120
 2121
 2122
 2123
 2124
 2125
 2126
 2127
 2128
 2129
 2130
 2131
 2132
 2133
 2134
 2135

004103' 000027' 000027
 004104' 010276' 000000*
 004105' 010300' 000000*
 004106' 010302' 000000*
 004107' 010304' 000000*
 004110' 010306' 000000*
 004111' 010310' 000000*
 004112' 010312' 000000*
 004113' 010314' 000364'
 004114' 010317' 000000*
 004115' 010321' 000000*
 004116' 010323' 000000*
 004117' 010325' 000000*
 004120' 010327' 000000*
 004121' 010331' 000000*
 004122' 010333' 004063*
 004123' 010337' 000000*
 004124' 010341' 000000*
 004125' 010343' 000000*
 004126' 010345' 000000*
 004127' 010347' 000000*
 004130' 010351' 000000*
 004131' 010353' 000000*
 004132' 010355' 000000*
 000030

;Here is the table that for the type command

TYPET:	\$STAB		;Start of the table
	KEYTAB	CTPPKT,CTPPKT	
	KEYTAB	DBUF1,DBUF1	;Routine to type out DBUF1
	KEYTAB	DBUF2,DBUF2	;Routine to type out DBUF2
	KEYTAB	DBUF3,DBUF3	;Routine to type out DBUF3
	KEYTAB	DBUF4,DBUF4	;Routine to type out DBUF4
	KEYTAB	DBUF5,DBUF5	;Routine to type out DBUF5
	KEYTAB	DBUF6,DBUF6	;Routine to type out DBUF6
	KEYTAB	MAPBBA,MAPPED-BUFFER	;Routine to type out MAPBB
	KEYTAB	MBUF1,MBUF1	;Routine to type out MBUF1
	KEYTAB	MBUF2,MBUF2	;Routine to type out MBUF2
	KEYTAB	MBUF3,MBUF3	;Routine to type out MBUF3
	KEYTAB	MBUF4,MBUF4	;Routine to type out MBUF4
	KEYTAB	MBUF5,MBUF5	;Routine to type out MBUF5
	KEYTAB	MBUF6,MBUF6	;Routine to type out MBUF6
	KEYTAB	RETBUF,RETURNED-BUFFER	
	KEYTAB	SAVCTP,SAVCTP	
	KEYTAB	SAVREQ,SAVREQ	
	KEYTAB	SAVRSP,SAVRSP	
	KEYTAB	SAVSCS,SAVSCS	
	KEYTAB	SCSCMD,SCSCMD	
	KEYTAB	SCSEVT,SCSEVT	
	KEYTAB	SCSRSP,SCSRSP	
	KEYTAB	SCSSAV,SCSSAV	
	\$ETAB		;End of type table

```

2136                                SUBTTL  Map-Buffer/Unmap-Buffer Commands
2137
2138                                ;Map a buffer
2139
2140 004133' 201 02 0 00 010363' .MAPB: GUIDE <number of bytes> ;Noise word
2141 004134' 260 17 0 00 004443'
2142 004135' 201 02 0 00 010372' MOVEI S2,[FLDDB.(.CMNUM,,^D10,<Bytes to map, 1-2048>)]
2143 004136' 260 17 0 00 004443' $CALL CMDPRS ;Parse the field please
2144 004137' 303 02 0 00 000000 CAILE S2,0 ;Skip if 0 or less
2145 004140' 303 02 0 00 004000 CAILE S2,^D2048 ;Skip if 2048 or less
2146 004141' 254 00 0 00 010412' CMDERR <Number of bytes to map not in range 0-2048> ;out of range
2147 004142' 202 02 0 00 000000* MOVEM S2,MAPBL ;Save it
2148 004143' 260 17 0 00 004435' $CALL CMDCFM ;We expect command confirmation here
2149
2150 004144' 402 00 0 00 003050* SETZM SCSERS ;Clear scs error indication
2151 004145' 260 17 0 00 000000* $CALL MPBUF ;Go map a buffer
2152 004146' 336 00 0 00 004144* SKIPN SCSERS ;Skip if no SCS error
2153 004147' 260 17 0 00 004073* $TEXT (,<Returned buffer name ^O/BUFRNM/>)
2154 004151' 263 17 0 00 000000 $RET
2155
2156                                ;Unmap A Buffer command
2157
2158 004152' 260 17 0 00 004435' .UMAPB: $CALL CMDCFM ;Command confirmation
2159 004153' 260 17 0 00 000000* $CALL UMPBUF ;Go unmap a buffer
2160 004154' 402 00 0 00 003021* SETZM BUFRNM ;Zero buffer name
2161 004155' 263 17 0 00 000000 $RET
  
```

```

2162          SUBTTL  PAUSE/LISTEN/ACCEPT Commands
2163
2164          ;PAUSE (for seconds) n
2165
2166          004156' 201 02 0 00 010434' .PAUSE: GUIDE <for seconds> ;Noise word
2167          004157' 260 17 0 00 004443'
2168
2169          004160' 201 02 0 00 010444'          MOVEI  S2,[FLDDB.(.CMNUM,,^D10,<Pause time in seconds, 1-2000>)]
2170          004161' 260 17 0 00 004443'          $CALL  CMDPRS          ;Parse it
2171          004162' 200 01 0 00 005505'          MOVE   S1,[^D1,,^D2000]          ;No, is this number in range, 1-2000 ?
2172          004163' 260 17 0 00 004417'          $CALL  RANGCK          ;Range check the number
2173          004164' 254 00 0 00 010465'          CMDERR <Number of seconds to pause out of range 1-2000> ;out of range
2174          004165' 260 17 0 00 004435'          $CALL  CMDCFM          ;We expect command confirmation here
2175          004166' 221 02 0 00 001750'          IMULI  S2,^D1000          ;Convert milliseconds to seconds
2176          004167' 200 01 0 00 000002'          MOVE   S1,S2          ;Put milli second wait time in S1
2177          004170' 260 17 0 00 000000*          $CALL  WAITX          ;Go to wait routine
2178          004171' 263 17 0 00 000000          $RET          ;Return to toplevel
2179
2180          ;LISTEN command
2181
2182          004172' 201 02 0 00 010475' .LISTE: GUIDE <for a connection>
2183          004173' 260 17 0 00 004443'
2184          004174' 260 17 0 00 004435'          $CALL  CMDCFM          ;Confirm the command
2185          004175' 254 00 0 00 000000*          JRST   LISTEN          ;Do the work
2186
2187          ;ACCEPT command
2188
2189          004176' 201 02 0 00 010502' .ACCEP: GUIDE <a connection>
2190          004177' 260 17 0 00 004443'
2191          004200' 260 17 0 00 004435'          $CALL  CMDCFM          ;Confirm the command
2192          004201' 254 00 0 00 000000*          JRST   ACCEPT          ;Do the work
  
```



```

2193          SUBTTL  DDT Command
2194
2195          ;DDT command, starts existing DDT or tries to merge it in and start it.
2196
2197 004202' 260 17 0 00 004435' .DDT:  $CALL  CMDCFM          ;Confirm command
2198
2199          ;Check for DDT loaded by LINK or for VMDDT (TOPS-10)
2200
2201 004203' 332 01 0 00 000000*      SKIPE  S1,,JBDDT          ;DDT present here?
2202 004204' 254 00 0 00 004242'      JRST   DDT3              ;Yes, DDT loaded by LINK or VMDDT (T10)
2203 004205' 336 00 0 00 000365'      SKIPN  MONTYP          ;TOPS-10?
2204 004206' 254 00 0 00 004233'      JRST   DDT1              ;Yep, DDT must not be there
2205
2206          ;Check for TOPS-20 UDDT, get it as needed
2207
2208 004207' 332 00 0 00 770000      SKIPE  770000          ;DDT present here?
2209 004210' 254 00 0 00 004223'      JRST   DDT6              ;Yes, UDDT is there (TOPS-20)
2210 004211' 205 01 0 00 100001      MOVX   S1,GJ%SHI!GJ%OLD    ;Old file short form
2211 004212' 561 02 0 00 010504'      HRROI  S2,[ASCIZ/SYS:UDDT.EXE/] ;File to look for
2212 004213' 104 00 0 00 000020      GTJFN%          ;Get a JFN for it
2213 004214' 320 16 0 00 004230'      ERJMP  DDT4              ;Punt
2214 004215' 505 01 0 00 400000      HRLI   S1,,FHSLF          ;Load this fork handle
2215 004216' 104 00 0 00 000200      GET%          ;Get the file
2216 004217' 320 16 0 00 004225'      ERJMP  DDT5              ;Punt
2217 004220' 201 01 0 00 400000      MOVEI  S1,,FHSLF          ;Load this fork handle
2218 004221' 200 02 0 00 010507'      MOVE   S2,[JRST SETUP]    ;Load the entry vector
2219 004222' 104 00 0 00 000204      SEVEC%          ;Set the entry vector
2220
2221 004223' 201 01 0 00 770000      DDT6:  MOVEI  S1,770000      ;Point to UDDT
2222 004224' 254 00 0 00 004242'      JRST   DDT3              ; and off we go
2223
2224 004225' 201 01 0 01 000000      DDT5:  MOVEI  S1,(S1)        ;Load just the JFN
2225 004226' 104 00 0 00 000022'      CLOSFX          ;Release the JFN
2226 004227' 320 16 0 00 004230'      ERJMP  .+1              ;Ignore errors
2227
2228          ;Here if no DDT and I can't load it
2229
2230 004230' 260 17 0 00 004147*      DDT4:  $TEXT  (,<? DDT not loaded>) ;Nope
2231 004232' 263 17 0 00 000000      R:      $RET              ;Return to toplevel
2232
2233          ;Here to try and merge in VMDDT on TOPS-10
2234
2235 004233' 202 17 0 00 000352'      DDT1:  MOVEM  P,PDL+PLEN-1    ;Save critical AC that MERGE. smashes
2236 004234' 201 01 0 00 010520'      MOVEI  S1,[EXP <SIXBIT/SYS/>,<SIXBIT/VMDDT/>,<SIXBIT/EXE/>,0,0,0]
2237 004235' 047 01 0 00 000173'      MERGE.  S1,              ;or CALLI ac,173
2238 004236' 334 17 0 00 000352'      SKIPA  P,PDL+PLEN-1    ;Owie, load P
2239 004237' 334 17 0 00 000352'      SKIPA  P,PDL+PLEN-1    ;OK, start me up
2240 004240' 254 00 0 00 004230'      JRST   DDT4              ;Owie
2241 004241' 201 01 0 00 700000      MOVEI  S1,700000          ;Point to VMDDT
2242
2243          ;Here to give message and start DDT
2244
2245 004242' 260 17 0 00 004230*      DDT3:  $TEXT  (,<To return, type R$G>)
2246 004244' 254 00 0 01 000000      JRST   (S1)              ;Enter DDT

```



```
2247          SUBTTL Request-Data Command
2248
2249      ;REQUEST-DATA command
2250
2251      004245' 201 02 0 00 010537' .DREQ: GUIDE <from> ;Noise word
2252      004246' 260 17 0 00 004443'
2253      004247' 201 02 0 00 010541' MOVE1 S2,[FLDDB.(.CMKEY,.DQSEN)]
2254      004250' 260 17 0 00 004443' $CALL CMDPRS ;Parse the field please
2255      004251' 550 01 0 02 000000 HRRZ S1,(S2) ;Set up to switch or retry routine
2256      004252' 260 17 0 00 004435' $CALL CMDCFM ;Confirm
2257      004253' 254 00 0 01 000000 JRST (S1) ;Go do the routine
2258
2259      ;Here is the table that for the request data command
2260
2261      004254' 000002 000002 DQSEN: $STAB ;Start of the request data table
2262      004255' 006651' 000000* KEYTAB RDATE,EXERCISER-NODE ;Routine to request data
2263      004256' 006654' 000000* KEYTAB RDATR,RESPONDER-NODE ;Routine to request data
2264      000003 $ETAB ;End of request data table
```

2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307

004257' 304 00 0 00 000000
 004263' 402 00 0 00 000001
 004264' 332 00 0 01 001436'
 004265' 254 00 0 00 004274'
 004266' 350 00 0 00 000001
 004267' 307 01 0 00 000017
 004270' 254 00 0 00 004264'
 004271' 260 17 0 00 004242*
 004273' 254 00 0 00 004276'

 004274' 202 01 0 00 000362'
 004275' 350 00 0 17 000000
 004276' 263 17 0 00 000000

 004277' 304 00 0 00 000000
 004303' 200 01 0 00 000362'
 004304' 254 00 0 00 004307'
 004305' 332 00 0 01 001436'
 004306' 254 00 0 00 004313'
 004307' 350 00 0 00 000001
 004310' 307 01 0 00 000017
 004311' 254 00 0 00 004305'
 004312' 254 00 0 00 004315'

 004313' 202 01 0 00 000362'
 004314' 350 00 0 17 000000
 004315' 263 17 0 00 000000

SUBTTL Get First Node/Next Node

 ;Get first node number from select node table. Call via 'GO GETFN'
 ;This subroutine will place the first node number in location NODEN
 ;and return +2. If no nodes were selected it will return +1.

 GETFN: \$SAVE <S1> ;Save some ACs
 SETZM S1 ;Zero the offset
 GETFN1: SKIPE SNODET(S1) ;Is this node selected ?
 JRST GETFN2 ; Yes
 AOS S1 ;Increment to next node entry
 CAIG S1,^D15 ;Have we tried all the entries
 JRST GETFN1 ; No
 \$TEXT (,<? No nodes are selected>) ; Yes. Take error return.
 JRST GETFN3

 ;There is a node in the selected table.
 GETFN2: MOVEM S1,NODEN ;Put it in location NODEN
 AOS (P) ;Adjust return address for success
 GETFN3: \$RET ;Return to calling

 ;Get next node number from select node table. Call via 'GO GETNN'
 ;This subroutine will place the next node number in location NODEN
 ;and return +2. If no more nodes are available to be tested it will
 ;return +1.

 GETNN: \$SAVE <S1> ;Save an AC
 MOVE S1,NODEN ;Get current node number
 JRST GETNN2 ;Go try the next node number
 GETNN1: SKIPE SNODET(S1) ;Is this node selected ?
 JRST GETNN3 ; Yes
 GETNN2: AOS S1 ;Increment to next node entry
 CAIG S1,^D15 ;Have we tried all the entries
 JRST GETNN1 ; No
 JRST GETNN4 ; Yes. Take error return.

 ;There is a node in the selected table.
 GETNN3: MOVEM S1,NODEN ;Put it in location NODEN
 AOS (P) ;Adjust return address for success
 GETNN4: \$RET ;Return to calling

```

2308          SUBTTL Logging Subroutines
2309
2310          ;LOGOPN - Subroutine to open the log file, LOGFB already set up.
2311
2312          LOGOPN: MOVEI S1,FOB.SZ          ;Load size of FOB
2313                  MOVEI S2,LOGFOB          ;Point to the FOB
2314                  $CALL F%AOPTN           ;Open up the thing append mode
2315                  JUMPF LOGOP2            ;Error.. punt it
2316                  MOVEM S1,LOGIFN         ;Save the IFN
2317                  $TEXT (LOUT%,<^I/ANNOUN/^F/LOGFD/ opened at ^H/[-1]/
2318                  >)                      ;Start the log file
2319                  $TEXT (T%TTY,<[^F/LOGFD/ opened]>);Output logging filename
2320                  $RET                    ;Return
2321
2322          LOGOP2: $TEXT (,<? ^E/S1/, logging not enabled>);Give message and return
2323                  $RET
2324
2325          ;Here to output a bell if needed
2326
2327          PBELL: SKIPN SWBELL              ;Bell set?
2328                  $RET                    ;Nope
2329                  $SAVE <S1>              ;Save an AC
2330                  MOVEI S1,'G'-100        ;Load a control-G
2331                  JRST K%BUFF             ;Dink bell to terminal only
2332
2333          ;PCRLF - output CRLF to terminal
2334
2335          PCRLF: $SAVE <S1>                ;Save an AC
2336                  HRROI S1,[ASCIZ/
2337                  /]                      ;Fall through to SOUT%%
2338
2339          ;SOUT%% - Routine that is called for string output to terminal.
2340          ; S1/ address of ASCIZ string
2341
2342          SOUT%%: $SAVE <P1,P2>            ;Save an AC
2343                  MOVEI P1,(S1)           ;Load the address of the string
2344                  HRLI P1,(Point 7)       ;Make a byte pointer out of it
2345
2346          SOUT%1: ILDB S1,P1               ;Load a byte
2347                  JUMPE S1,.POPJ          ;Jump if done
2348                  $CALL BOUT%%            ;Output the byte to the file
2349                  JRST SOUT%1             ;Loop for all bytes in the string
  
```

```

2350
2351 ;BOUT%% - Routine that is called for printing a character in S1.
2352 ; Sends character to terminal and log file if opened.
2353
2354 004357' 332 00 0 00 000371' BOUT%%: SKIPE LOGIFN ;Skip if no logging IFN
2355 004360' 260 17 0 00 004365' $CALL LOUT%% ;Output to log file
2356 004361' 260 17 0 00 004342* $CALL K%BUFF ;Buffer output to file
2357 004362' 306 01 0 00 000012 CAIN S1,.CHLFD ;Linefeed?
2358 004363' 260 17 0 00 000000* $CALL K%FLSH ;Yes, kawhoosh
2359 004364' 263 17 0 00 000000 $RET ;Return
2360
2361 ;LOUT%% - Routine to output character to LOG file, S1/ character
2362
2363 004365' 304 00 0 00 000000 LOUT%%: $SAVE <S1,S2> ;Save the ACs in question
2364 004372' 200 02 0 00 000001 MOVE S2,S1 ;Copy byte to S2
2365 004373' 336 01 0 00 000371' SKIPN S1,LOGIFN ;Load the IFN
2366 004374' 260 17 0 00 000000* $STOP (LOC,Output to Log file when not opened)
2367 004376' 260 17 0 00 000000* $CALL F%OBYT ;Output the byte
2368 004377' 326000 004354* $RETI ;Return if OK
2369 004400' 260 17 0 00 004330* $TEXT (T%TTY,<? Output error to LOG file: ^E/S1/>)
2370 ;Fall thru to close LOG file
2371
2372 ;LOGCLS - Suboroutine to close the log file, checks if it was open first
2373
2374 004402' 336 01 0 00 000371' LOGCLS: SKIPN S1,LOGIFN ;Skip and load existing IFN
2375 004403' 263 17 0 00 000000 $RET ;Return now
2376 $TEXT (LOUT%%,<
2377 004404' 260 17 0 00 004400* ^F/LOGFD/ closed at ^H/[-1]/>)
2378 004406' 260 17 0 00 003167* $CALL F%REL ;Release the IFN
2379 004407' 402 00 0 00 000371' SETZM LOGIFN ;Clear that IFN
2380 004410' 336 00 0 00 000000 SKIPN S1,LOGIFN ;Skip if it went well
2381 004411' 260 17 0 00 004404* $TEXT (,<? ^F/LOGFD/ close error: ^E/S1/>)
2382 004413' 332 00 0 00 000000 SKIPF ;Skip if it didn't go so well
2383 004414' 260 17 0 00 004411* $TEXT (,<[^F/LOGFD/ closed]>) ;Close message
2384 004416' 263 17 0 00 000000 $RET ;Return OK

```

SUBTTL Parsing Subroutines

;RANGCK -- Range test a number
 ;Multiple returns are supplied for branching.

;CALLING SEQ:

MOVE	S1,[LIM1,,LIM2]	;Specify the limits
MOVE	S2,NUM	;Number to be tested
\$CALL	RANGCK	;Call the routine
\$RET-1		;Out of range
\$RET-2		;Normal return

RANGCK: \$SAVE <T1,T2> ;Save some ACs

;Set up 36 bit limits with lower of the 2 first on the stack

HLRE	T1,S1	;Get low range
HRRE	T2,S1	;Get high range
CAML	T1,T2	;Low one lower?
EXCH	T1,T2	;Nope, fix

;Range check a number

CAML	S2,T1	;Is num at least lower limit ?
CAMLE	S2,T2	;Yes, is it within upper limit ?
\$RET		;No, out of range
AOS	(P)	;Bump the return
\$RET		;Return to call

2385
 2386
 2387
 2388
 2389
 2390
 2391
 2392
 2393
 2394
 2395
 2396
 2397 004417' 304 00 0 00 000000
 2398
 2399
 2400
 2401 004424' 574 03 0 00 000001
 2402 004425' 570 04 0 00 000001
 2403 004426' 311 03 0 00 000004
 2404 004427' 250 03 0 00 000004
 2405
 2406
 2407
 2408 004430' 311 02 0 00 000003
 2409 004431' 313 02 0 00 000004
 2410 004432' 263 17 0 00 000000
 2411 004433' 350 00 0 17 000000
 2412 004434' 263 17 0 00 000000


```

2413
2414 ;CMDCFM - subroutine to parse a confirm.
2415
2416 004435' 304 00 0 00 000000' CMDCFM: $SAVE <S1,S2> ;Save 2 ACs
2417 004442' 201 02 0 00 001540' MOVEI S2,CFM ;Just a confirm please
2418 ;Fall thru to parse it
2419
2420 ;CMDPRS - Come here to parse a command with function block in S2.
2421 ;Returns S2/result, T1/PDB used, S1/flags
2422
2423 004443' 201 01 0 00 000400' CMDPRS: MOVEI S1,CSB ;Point to the CSB
2424 004444' 260 17 0 00 000000* $CALL S%CMND ;Hello GLXLIB
2425 004445' 322 00 0 00 004470' JUMPF PARERR ;Pass along a false return
2426 004446' 200 01 0 02 000000 MOVE S1,CR.FLG(S2) ;Get flag word of reply block
2427 004447' 603 01 0 00 200000 TXNE S1,CM%NOP ;Was no-parse set?
2428 004450' 254 00 0 00 004470' JRST PARERR ;Yes
2429 004451' 550 03 0 02 000002 HRRZ T1,CR.PDB(S2) ;Load the PDB used
2430 004452' 200 02 0 02 000001 MOVE S2,CR.RES(S2) ;Load the result
2431 004453' 302 03 0 00 001540' CAIE T1,CFM ;Was it the confirm one
2432 004454' 263 17 0 00 000000 $RET ;No, return to caller
2433
2434 ;CMDLOG - Log a the contents of the command parse buffer
2435
2436 004455' 260 17 0 00 004350* CMDLOG: $SAVE <P1,P2> ;Save a couple
2437 004456' 200 07 0 00 000404' MOVE P1,CSB+.CMPTR ;Copy buffer pointer so won't
2438 004457' 400 10 0 00 000000 SETZ P2, ;distrub it when we
2439 004460' 136 10 0 00 000007 IDPB P2,P1 ;insure a null at end of it
2440 004461' 332 00 0 00 000371' SKIPE LOGIFN ;Skip if not logging
2441 004462' 260 17 0 00 004414* $TEXT (LOUT%,<^T/PROMPT/^T/BUFFER/^A>)
2442 004464' 332 00 0 00 000372' SKIPE TAKIFN ;Skip if not taking
2443 004465' 201 17 0 00 004462* $TEXT (T%TTY,<^T/PROMPT/^T/BUFFER/^A>)
2444 004467' 263 17 0 00 000000 $RET ;Return
2445
2446 ;PARERR - Come here on parse error to print standard message and return.
2447
2448 004470' 260 17 0 00 004455' PARERR: $CALL CMDLOG ;Log the failing command
2449 004471' 260 17 0 00 003172' $CALL TAKEOF ;Check for take command file EOF
2450 004472' 326 00 0 00 004506' JUMPT PARER3 ;Jump if EOF
2451
2452 004473' 260 17 0 00 000000* PARER1: $CALL S%ERR ;Load error text pointer to S1
2453 004474' 336 00 0 00 000000 SKIPT ;Skip if it was OK
2454 004475' 201 01 0 00 010743' MOVEI S1,[ASCIZ/Command error/] ;No? Load a default message
2455 004476' 260 17 0 00 004455' $CALL CMDLOG ;Log the command
2456 004477' 260 17 0 00 004465' $TEXT (<? ^T/(S1)/, type HELP for help>) ;Publish message
2457 004501' 336 00 0 00 000372' SKIPN TAKIFN ;Skip if in a take
2458 004502' 254 00 0 00 002020' JRST TOPLVL ;No, restart everything
2459 004503' 260 17 0 00 004477* $TEXT (<? Aborting TAKE file ^F/TAKFD/>) ;Give me a sign, o Leader
2460 004505' 254 00 0 00 003161' JRST TAKABT ;Abort it and return
2461
2462 004506' 336 00 0 00 000372' PARER3: SKIPT TAKIFN ;Skip if a TAKE IFN there
2463 004507' 254 00 0 00 004473' JRST PARER1 ;Nope
2464 $TEXT (<
2465 004510' 260 17 0 00 004503* [End of ^F/TAKFD/]>) ;End of take file
2466 004512' 254 00 0 00 003161' JRST TAKABT ;Abort take file and return

```

```

2467          SUBTTL  ERRHAN - Produce Error Messages
2468
2469          ;*****
2470          ;Subroutine to report errors from ERROR1 macro
2471          ;Call with EM pointing to block:
2472          ;0      X      ;Error message output control
2473          ;1      CORR   ;Addr of correct data if X<>0
2474          ;2      ACTU   ;Addr of actual data if X<>0
2475          ;3      [ASCIZ\FUNCT\] ;Addr of ASCIZ function or 0
2476          ;4      [ASCIZ\CMNT\] ;Addr of comment line or 0
2477          ;5      ROUT   ;Addr of routine to call or 0
2478          ;Stack contains Error PC address
2479          ;*****
2480
2481          004513' 260 17 0 00 004333'   ERRHAN: $CALL  PBELL           ;Print a bell if appropriate
2482          $TEXT  (<
2483          ? Error detected at ^H/[-1]/ Pass ^D/PASCNT/. Iteration ^D/ITCNT/.
2484          ^T/@TSTNAM/ - ^T/@TSTDSC/>)
2485
2486          ;Output function tested (if present).
2487
2488          004516' 332 00 0 13 000003     $KIPE  3(EM)           ;Any particular function?
2489          004517' 260 17 0 00 004514*   $TEXT  (< ^T/@3(EM)/>) ;Yes
2490
2491          ;Output actual and correct numbers (if present).
2492
2493          004521' 333 00 0 13 000000     $KIPE  0(EM)           ;Any octal error stuff to print?
2494          $TEXT  (< Correct: ^O/@1(EM)/
2495          Actual: ^O/@2(EM)/>)
2496          004522' 260 17 0 00 004517*   $SKIPGE 0(EM)         ;Any decimal error stuff to print?
2497          004524' 335 00 0 13 000000     $TEXT  (< Correct: ^D/@1(EM)/
2498          Actual: ^D/@2(EM)/.>)
2499
2500          ;Output comment text if present.
2501
2502          004527' 332 00 0 13 000004     $KIPE  4(EM)           ;Any diag comment
2503          004530' 260 17 0 00 004525*   $TEXT  (< ^T/@4(EM)/>) ;Yes
2504
2505          ;Call routine to output additional information if present.
2506
2507          004532' 332 00 0 13 000005     $KIPE  5(EM)           ;Any routine to call?
2508          004533' 260 17 1 13 000005     $CALL  @5(EM)         ;Yep
2509
2510          ;Output extra CRLF and check if halting on error, if not return.
2511
2512          004534' 260 17 0 00 004343'   $CALL  PCRLF           ;Output crlf and return
2513          004535' 336 00 0 00 000376'   $SKIPN  SWHALT        ;Halt on error?
2514          004536' 263 17 0 00 000000     $RET                    ;Nope, continue
2515          004537' 260 17 0 00 002652'   $CALL  UNLKIT         ;Unlock this image
2516          004540' 254 00 0 00 002020'   JRST   TOPLVL         ;Return to top level parser

```

```

2517          SUBTTL Run Time Information
2518
2519          ;Here to see if the user wants to know the current status, called from
2520          ;anywhere there is a loop.
2521
2522          004541' 304 00 0 00 000000          RTCHK: $SAVE    <S1,S2,T1,T2>          ;Save some ACs
2523          004550' 336 00 0 00 000365'          SKIPN    MONTP          ;Twenty?
2524          004551' 254 00 0 00 004560'          JRST     RTCHK1          ;Ten
2525
2526          ;Here if tops orange to get a character from the terminal if any
2527
2528          004552' 201 01 0 00 000100          MOVEI     S1,..PRIIN          ;Load primary input designator
2529          004553' 104 00 0 00 000102          SIBEX          ;Skip if input buffer empty
2530          004554' 304 00 0 00 000000          CAIA          ;Not empty
2531          004555' 263 17 0 00 000000          $RET          ;Empty, return now
2532          004556' 104 00 0 00 000073'          PBINX          ;Its not empty, get a character
2533          004557' 254 00 0 00 004562'          JRST     RTCHK2          ;Got one, examine it
2534
2535          ;Here if tops blue to get a character if any typed
2536
2537          004560' 051 02 0 00 000001          RTCHK1: INCHRS  S1          ;Skip if a character can be input
2538          004561' 263 17 0 00 000000          $RET          ;No character there
2539
2540          ;Check input character, do requested function
2541
2542          004562' 301 01 0 00 000141          RTCHK2: CAIL     S1,'a'          ;Is it lower case?
2543          004563' 275 01 0 00 000040          SUBI     S1,'a'-'A'          ;Yes convert to upper
2544          004564' 205 02 0 00 777774          MOVS    S2,-RTTABL          ;Load AOB pointer to table
2545
2546          004565' 554 03 0 02 004575'          RTCHK3: HLRZ     T1,RTTAB(S2)          ;Load a character
2547          004566' 302 01 0 03 000000          CAIE     S1,(T1)          ;Match status request?
2548          004567' 253 02 0 00 004565'          AOBJN    S2,RTCHK3          ;Nope, keep looking
2549          004570' 550 01 0 02 004575'          HRRZ     S1,RTTAB(S2)          ;(Possible) match, load address
2550          004571' 321 02 0 01 000000          JUMPL    S2,(S1)          ;Dispatch if character match
2551          $TEXT    (<
2552          004572' 260 17 0 00 004530*          ? Type H for help>)          ;Message
2553          004574' 263 17 0 00 000000          $RET          ;Return
2554
2555          ;Table of things to do
2556
2557          004575' 000033 004613'          RTTAB:  XWD     ,CHESC,RTCHK$          ;Abort
2558          004576' 000110 004601'          XWD     ,H,RTCHKH          ;Help
2559          004577' 000123 004617'          XWD     ,S,RTCHK$          ;Status
2560          004600' 000124 004604'          XWD     ,T,RTCHKT          ;Trace toggle
2561          RTTABL==.-RTTAB          ;Length of the table

```

```
2562
2563 ;Here for help
2564
2565 RTCHKH: $TEXT (,<
2566 Run time command options:
2567 H For this message
2568 S To get current test status
2569 T Toggle the TRACE switch
2570 004601' 260 17 0 00 004572* esc Escape to abort testing>)
2571 004603' 263 17 0 00 000000 $RET
2572
2573 ;Here on a 'T'
2574
2575 004604' 462 00 0 00 000374' RTCHKT: SETCMM SWTRAC ;Complement trace switch
2576 004605' 336 00 0 00 000374' SKIPN SWTRAC ;Trace set?
2577 004606' 334 01 0 00 011176' SKIPA S1,[ASCIZ/off/] ;No
2578 004607' 205 01 0 00 677340 MOVSI S1,(ASCIZ/on/) ;Yes
2579 $TEXT (,<
2580 004610' 260 17 0 00 004601* Trace mode is now ^T/S1/>)
2581 004612' 263 17 0 00 000000 $RET ;Return
2582
2583 ;Here on an escape.
2584
2585 RTCHK$: $TEXT (,<
2586 004613' 260 17 0 00 004610* Aborted testing during ^T/@TSTNAM/ pass ^D/PASCNT/ iteration ^D/ITCNT/ on node ^D/NODEN/.)>
2587 004615' 260 17 0 00 002652' $CALL UNLKIT ;Unlock me
2588 004616' 254 00 0 00 002020' JRST TOPLVL ;Restart to parser
2589
2590 ;Here on a 'S'.
2591
2592 RTCHKS: $TEXT (,<
2593 004617' 260 17 0 00 004613* ^T/@TSTNAM/ pass ^D/PASCNT/ iteration ^D/ITCNT/ on node ^D/NODEN/.)>
2594 004621' 263 17 0 00 000000 $RET ;Return
```



```
2595          SUBTTL Literals
2596
2597          ;Here is the literal pool for this module.
2598
2599 004622'    LIT..P: XLIST          ;Lit
2600          LIST
2601
2602          001672'          END      SETUP
```

NO ERRORS DETECTED

PROGRAM BREAK IS 011350
CPU TIME USED 02:20.193

172P CORE USED

ACCEPT	140#	2192					
ACTFLG	136#	1849					
ANNOUN	638#						
ATMBUF	171	272#					
ATMSZ	172	272#	277				
BACTF	1807	1842#					
BALL	153#	427					
BBGENL	1741	1749	1762	1933#			
BDELAY	1756	1773	1785	1795	1963#		
BEXTEN	1787	1797	1863#				
BFLNM1	136#						
BGENC	1740	1776	1923#				
BGENF	1739	1775	1913#				
BIDATA	1778	2003#					
BLDCTP	133#	1728					
BLOFFS	1763	1983#					
BMLEN	1777	1943#					
BMOVT3	1758	1873#					
BOTHER	1760	1788	1798	1814	1821	1828	1903#
BOUT%	160	2348	2354#				
BPKTM	1761	1893#					
BPKTS	1759	1883#					
BRCNT	1757	1774	1786	1796	1973#		
BROFFS	1764	1993#					
BSTADR	1800	1953#					
BTYPE	1738	1748	1832#				
BU0	1727	1733#					
BU1	1727	1737#					
BU10	1727	1820#					
BU11	1727	1827#					
BU12	1727	1733#					
BU13	1727	1733#					
BU2	1727	1747#					
BU3	1727	1755#					
BU4	1727	1772#					
BU5	1727	1771#					
BU6	1727	1784#					
BU7	1727	1794#					
BU8	1727	1806#					
BU9	1727	1813#					
BUFALL	153#	428					
BUFFER	168	169	271#				
BUFLEN	133#	1940					
BUFLNM	136#						
BUFRNM	136#	1380	1382	2160			
BUFSIZ	170	271#					
BUFTYP	134#	1839					
BYTE1	2075#						
BYTE2	2076#						
BYTE3	2077#						
BYTE4	2078#						
BYTE5	2079#						
C..CLE	298#	490					

GETNN4	2301	2307#														
GJ%FOU	788															
GJ%OLD	1457	2210														
GJ%SHT	2210															
GJFBLK	174	273#	780	781	782	789	791	793	795	797	814	816	818	821		
	1447	1448	1449	1458	1460	1462	1464	1466	1484	1486	1488	1491				
GJFSIZ	273#	782	822	1449	1492											
GLNN	126#	1320														
GNCNST	134#	1930														
GPRS	128#	1415														
GPRSB	128#	1418	1422													
GS%EOF	1529															
HAL	481	484#	1036													
HCS	475#															
HELPO	927#	929														
HELP1	924	933	940#													
HELP2	951	954	961#	979	986											
HELP3	967#															
HELP6	968	983#														
HELP7	962	990#														
HFIND	950	953	956	1060#												
HFOB10	218#	1064														
HFOB20	206#	1063														
HLPADD	490#	493	927	930												
HLPADL	493#	926														
HLPARS	474#	921														
HLPAT1	1041#	1046														
HLPAT3	1035	1049#														
HLPAT4	1037	1053#														
HLPATM	923	932	1034#													
HLPBUF	277#	919	920	942	944	970	973	975	1054							
HLPEOF	279#	940	961	971	1010											
HLPERR	957	1073#														
HLPFD1	219	223#	1060													
HLPFD2	207	211#	1061													
HLPIFN	280#	992	997	1068												
HLPLIN	278#	972	983	998												
HREAD	963	997#	1020													
HREAD1	1007#	1014	1018													
HREAD2	1000	1010#														
HREAD3	1008	1013#														
HREAD4	1016	1020#														
HREAD5	1011	1022	1028#													
HSC	475	478#														
HTL	478	481#														
IXEXIT	1553															
IXINIT	560															
IB	158#	559														
IB.FLG	161															
IB.OUT	160															
IB.PRG	159															
IB.SZ	158	558														
IMGDAT	133#	2009														

SEQ 0318

SEQ 0320

[illegible]

[illegible]

SETUP1	588	601#																	
SETUP3	565	573#																	
SEVNT	126#	1439																	
SHALL	337	1280#																	
SHCICO	338	1367#																	
SHCID	341	1373#																	
SHCO	339	1396#																	
SHCST	126#	1403																	
SHEVNT	340	1438#																	
SHMBN	343	1379#																	
SHMBS	344	1388#																	
SHOTAB	336#	492	1273																
SHPOLL	126#	1433																	
SHPS	346	1408#																	
SHPST	347	1432#																	
SHRTBL	348	1349#																	
SHSN	349	1325#																	
SHSN1	1328#	1333																	
SHSN2	1329	1333#																	
SHSOCN	350	1402#																	
SHSPS	345	1291#																	
SHSPS1	1293#	1295																	
SHSW1	1305	1309#																	
SHSW2	1310#	1314																	
SHSWI	351	1302#																	
SHTIOU	352	1340#																	
SL3B	759	768#	1292	1293	1296														
SLNODE	342	1319#																	
SNALL	379	382#	836	888															
SNDDAT	136#	1582																	
SNDMSG	133#	1576																	
SNODET	118	288#	851	860	870	896	905	911	912	913	1328	2273	2296						
SNTAB	383	384#																	
SOUT%	985	2342#																	
SOUT%1	2346#	2349																	
SRTB2	1356#	1360																	
STADR	133#	1960																	
STRB3	1352	1362#																	
SWBELL	266#	370	2327																
SWHALT	265#	371	2513																
SWITAB	369#	374	686	714	1310	1311													
SWITCN	374#	1309																	
SWPALL	264#	372																	
SWTRAC	119	263#	373	2575	2576														
T%TEXT	582	593	603	606	609	701	734	749	777	840	846	852	871	892					
	897	991	1073	1106	1139	1155	1177	1253	1264	1297	1303	1306	1313	1331					
	1332	1335	1344	1354	1358	1362	1374	1381	1383	1391	1413	1427	1446	1502					
	1507	1725	1733	1838	1848	1869	1879	1889	1899	1909	1919	1929	1939	1949					
	1959	1969	1979	1989	1999	2008	2096	2101	2146	2153	2173	2230	2245	2278					
	2318	2319	2322	2369	2377	2381	2383	2441	2443	2456	2459	2465	2484	2489					
	2495	2498	2503	2552	2570	2580	2586	2593											
T1	595	692	720	803	836	857	888	902	926	927	929	930	967	1002					
	1003	1020	1021	1034	1036	1041	1042	1043	1044	1045	1095	1098	1113	1129					

11

	1131	1147	1165	1167	1311	1351	1352	1356	1357	1472	1737	1747	1755	1772
	1784	1794	1806	1813	1820	1827	2094	2099	2104	2397	2401	2403	2404	2408
T2	2429	2431	2522	2546	2547									
	927	928	998	1005	1017	1024	1026	1029	1309	1310	1311	1314	2397	2402
	2403	2404	2409	2522										
T3	920	945	978	990	1304	1307	1312							
T4	919	936	941	945	969	978	984	1044	1050	1140	1141	1183	1187	1188
	1189	1229	1304	1307	1310	1341	1737	1747	1755	1772	1784	1794	1806	1813
	1820	1827	2271	2293	2329	2335	2342	2363	2397	2416	2436	2522		
TAKABT	1508	1512#	2460	2466										
TAKE1	1453	1483#												
TAKE3	1479	1502#												
TAKE6	1478	1497	1507#											
TAKEO1	1525	1533#												
TAKEOF	1524#	2449												
TAKFD	184	188#	1470	1491	1492									
TAKFOB	183#	1495												
TAKIFN	260#	657	1445	1471	1475	1498	1512	1519	1526	1533	2442	2457	2462	
TALL	153#	426												
TITCNT	119	248#	1207	1233										
TOPLVL	597	610	651#	701	734	749	777	840	846	852	892	897	1106	1139
	1155	1415	1446	1513	1520	1725	1733	1838	1848	1869	1879	1889	1899	1909
	1919	1929	1939	1949	1959	1969	1979	1989	1999	2008	2146	2173	2458	2516
	2588													
TSPAR1	449	452#												
TSPAR2	440	443#												
TST01	148#	391												
TST02	148#	392												
TST10	149#	394												
TST11	149#	395												
TST12	149#	396												
TST13	149#	397												
TST14	149#	398												
TST30	150#	400												
TST31	150#	401												
TST32	150#	402												
TST33	150#	403												
TST40	150#	404												
TST50	151#	406												
TST51	151#	407												
TST52	151#	408												
TST53	151#	409												
TST54	151#	410												
TST55	151#	411												
TST56	151#	412												
TST57	151#	413												
TST58	151#	414												
TST59	151#	415												
TST80	152#	417												
TST99	144#	419	1185	1236										
TSTDSC	119	244#												
TSTNAM	119	243#												
TSTTAB	390#	444	453	482										

[illegible]

..0037	1641#	1644												
..0040	1641	1644#												
..0041	1660#	1665												
..0042	1660	1665#												
..0043	1677#	1690												
..0044	1677	1690#												
..0045	1702#	1716												
..0046	1702	1716#												
..0047	1738	1738#												
..0050	1748	1748#												
..0051	1756	1756#												
..0052	1773	1773#												
..0053	1785	1785#												
..0054	1795	1795#												
..0055	1807	1807#												
..0056	1814	1814#												
..0057	1821	1821#												
..0060	1828	1828#												
..0061	2025#	2028												
..0062	2025	2028#												
..0063	2053#	2056												
..0064	2053	2056#												
..0065	2111#	2135												
..0066	2111	2135#												
..0067	2261#	2264												
..0070	2261	2264#												
..0071	2272	2272#												
..0072	2294	2294#												
..0073	2330	2330#												
..0074	2336	2336#												
..0075	2343#													
..0076	2364	2364#												
..0077	2398	2398#												
..0100	2417	2417#												
..0101	2437#													
..0102	2523	2523#												
..BLOC	158#	162	163	166#	175	176	183#	186	187	188#	190	191	195#	198
	199	200#	202	203	206#	209	210	211#	216	217	218#	221	222	223#
..BSIZ	228	229												
	158	158#	159	160	161	162	163	166	166#	167	168	169	170	171
	172	173	174	175	176	183	183#	184	185	186	187	188	188#	189
	190	191	195	195#	196	197	198	199	200	200#	201	202	203	206
	206#	207	208	209	210	211	211#	212	213	214	215	216	217	218
..MX1	218#	219	220	221	222	223	223#	224	225	226	227	228	229	
	607#	607	608	788#	788	789	803#	803	804	1457#	1457	1458	1476#	1476
..MX2	1477	2210#	2210	2211										
	607#	607	608	788#	788	789	803#	803	804	1457#	1457	1458	1476#	1476
..T	1477	2210#	2210	2211										
	163#	163	176#	176	187#	187	191#	191	199#	199	203#	203	210#	210
..TO	217#	217	222#	222	229#	229								
	158#	160	160#	163	166#	167	167#	176	183#	184	184#	187	188#	189
	189#	191	195#	196	196#	199	200#	201	201#	203	206#	207	207#	210
	211#	212	212#	217	218#	219	219#	222	223#	224	224#	229		

[illegible]

..T62	188#	191	200#	203										
..T63	188#	191	200#	203										
..T64	188#	191	200#	203										
..T65	188#	191	200#	203										
..T66	188#	191	200#	203										
..T67	188#	191	200#	203										
..T7	166#	171	171#	176	188#	191	200#	203						
..T70	188#	191	200#	203										
..T71	188#	191	200#	203										
..T72	188#	191	200#	203										
..T73	188#	191	200#	203										
..TMSK	159#	159	160#	160	161#	161	167#	167	168#	168	169#	169	170#	170
	171#	171	172#	172	173#	173	174#	174	184#	184	185#	185	189#	189
	196#	196	197#	197	201#	201	207#	207	208#	208	212#	212	213#	213
	214#	214	215#	215	219#	219	220#	220	224#	224	225#	225	226#	226
	227#	227												
..TSA1	1229#	1229	1341#	1341	1737#	1737	1747#	1747	1755#	1755	1772#	1772	1784#	1784
	1794#	1794	1806#	1806	1813#	1813	1820#	1820	1827#	1827	2271#	2271	2293#	2293
	2329#	2329	2335#	2335	2342#	2342	2363#	2363	2397#	2397	2416#	2416	2436#	2436
	2522#	2522												
..TVAL	159#	159	160#	160	161#	161	167#	167	168#	168	169#	169	170#	170
	171#	171	172#	172	173#	173	174#	174	184#	184	185#	185	189#	189
	196#	196	197#	197	201#	201	207#	207	208#	208	212#	212	213#	213
	214#	214	215#	215	219#	219	220#	220	224#	224	225#	225	226#	226
	227#	227												
..TX1	547#	547	548	1529#	1529	1530	2427#	2427	2428					
..TX2	547#	548	1529#	1529	1530	2427#	2427	2428						
..TXB	2367#	2367												
..TXC	2367#	2367												
..TXEF	2367#	2367												
..TXEG	2367#	2367												
..TXF	2367#	2367												
..TXP	2367#	2367												
..XX	357#	357	379#	379	382#	382	440#	440	443#	443	447#	447	449#	449
	452#	452	456#	456	459#	459	475#	475	478#	478	481#	481	484#	484
	490#	490	491#	491	492#	492	663#	663	668#	668	679#	679	686#	686
	690#	690	707#	707	714#	714	718#	718	728#	728	730#	730	743#	743
	745#	745	759#	759	778#	778	798#	798	819#	819	831#	831	854#	854
	856#	856	883#	883	899#	899	901#	901	1088#	1088	1103#	1103	1136#	1136
	1145#	1145	1271#	1271	1273#	1273	1408#	1408	1410#	1410	1450#	1450	1467#	1467
	1489#	1489	1548#	1548	1558#	1558	1560#	1560	1587#	1587	1589#	1589	1618#	1618
	1620#	1620	1634#	1634	1648#	1648	1669#	1669	1694#	1694	1720#	1720	1722#	1722
	1832#	1832	1834#	1834	1842#	1842	1844#	1844	1863#	1863	1865#	1865	1873#	1873
	1875#	1875	1883#	1883	1885#	1885	1893#	1893	1895#	1895	1903#	1903	1905#	1905
	1913#	1913	1915#	1915	1923#	1923	1926#	1926	1933#	1933	1935#	1935	1943#	1943
	1945#	1945	1953#	1953	1955#	1955	1963#	1963	1965#	1965	1973#	1973	1975#	1975
	1983#	1983	1985#	1985	1993#	1993	1995#	1995	2003#	2003	2005#	2005	2015#	2015
	2017#	2017	2045#	2045	2083#	2083	2085#	2085	2140#	2140	2142#	2142	2166#	2166
	2169#	2169	2182#	2182	2189#	2189	2251#	2251	2253#	2253				
..A	1229	1341	1737	1747	1755	1772	1784	1794	1806	1813	1820	1827	2271	2293
..A1	2329	2335	2343	2363	2397	2416	2437	2522						
	1229	1341	1737	1747	1755	1772	1784	1794	1806	1813	1820	1827	2271	2293
	2329	2335	2343	2363	2397	2416	2437	2522						

.A15	1229	1341	1737	1747	1755	1772	1784	1794	1806	1813	1820	1827	2271	2293
.A16	2329	2335	2343	2363	2397	2416	2437	2522						
.ACB	1229	1341	1737	1747	1755	1772	1784	1794	1806	1813	1820	1827	2271	2293
	2329	2335	2342	2343	2363	2397	2416	2436	2437	2522				
	1229#	1229	1341#	1341	1342#	1342	1737#	1737	1738#	1738	1747#	1747	1748#	1748
	1755#	1755	1756#	1756	1772#	1772	1773#	1773	1784#	1784	1785#	1785	1794#	1794
	1795#	1795	1806#	1806	1807#	1807	1813#	1813	1814#	1814	1820#	1820	1821#	1821
	1827#	1827	1828#	1828	2271#	2271	2272#	2272	2293#	2293	2294#	2294	2329#	2329
	2330#	2330	2335#	2335	2336#	2336	2342#	2342	2363#	2363	2364#	2364	2397#	2397
	2398#	2398	2416#	2416	2417#	2417	2436#	2436	2522#	2522	2523#	2523		
.ACCEP	296	2189#												
.ACM	1229#	1229	1341#	1341	1342	1737#	1737	1738	1747#	1747	1748	1755#	1755	1756
	1772#	1772	1773	1784#	1784	1785	1794#	1794	1795	1806#	1806	1807	1813#	1813
	1814	1820#	1820	1821	1827#	1827	1828	2271#	2271	2272	2293#	2293	2294	2329#
	2329	2330	2335#	2335	2336	2342#	2342	2343	2363#	2363	2364	2397#	2397	2398
	2416#	2416	2417	2436#	2436	2437	2522#	2522	2523					
.APRID	630													
.BUILD	297	1720#												
.CHCRT	1013	1023												
.CHESC	2557													
.CHLFD	1015	1025	2357											
.CLEAR	298	679#												
.CLOG	363	700#												
.CMABC	172													
.CMABP	171													
.CMBFP	168													
.CMCFM	357													
.CMCMA	456	690	718	854	899	1145								
.CMCNT	170													
.CMFIL	798	1467												
.CMFLG	167													
.CMGJB	166	174	270	653										
.CMIFI	1489													
.CMINI	663													
.CMIOJ	553	659	790	1459										
.CMKEY	382	440	443	449	452	475	478	481	490	491	492	668	679	686
	707	714	759	1273	1560	1589	1620	1634	1648	1669	1694	2017	2045	2085
	2253													
.CMNOI	728	743	778	831	883	1088	1271	1408	1450	1548	1558	1587	1618	1720
	1832	1842	1863	1873	1883	1893	1903	1913	1923	1933	1943	1953	1963	1973
	1983	1993	2003	2015	2083	2140	2166	2182	2189	2251				
.CMNUM	379	730	745	856	901	1103	1136	1410	1722	1834	1844	1865	1875	1885
	1895	1905	1915	1926	1935	1945	1955	1965	1975	1985	1995	2005	2142	2169
.CMOFI	819													
.CMPTR	169	2437												
.CMRTY	173	661												
.CMSWI	447	459												
.CMTOK	484													
.CONN	299	1605#												
.CPUTY	614													
.DATA	1568	1587#												
.DCONN	302	1611#												
.DDT	300	2197#												

SEQ 0329

.SDATA	1569	1581#	
.SECIA	1706		
.SECRA	1704		
.SECRR	1705		
.SECTL	1703		
.SELCL	1707		
.SELEC	314	831#	
.SELNO	835#	858	
.SELN1	850#		
.SELNA	385	837	865#
.SEMDC	1708		
.SEMSC	1709		
.SESCO	1710		
.SEND	315	1558#	
.SENWO	1711		
.SEOSD	1712		
.SEPBC	1713		
.SERID	1714		
.SET	316	707#	
.SEVCC	1715		
.SHOW	317	1271#	
.SHSN	1283	1326#	
.SHSPS	1282	1292#	
.SHSWI	1286	1303#	
.SL1	331	728#	
.SL2	330	743#	
.SL3	328	759#	
.SLNA1	867#	874	
.SLNA2	869	872#	
.SLOG	327	776#	
.SLOG5	784	813#	
.SMESS	1570	1575#	
.SNODE	1281	1320#	
.SPS1	1417	1428#	
.SPST	1433#		
.SQRPS	1418	1422	
.SRTBL	1285	1350#	
.SSAPS	771		
.SSOCN	1403#		
.SSPTA	769		
.SSPTB	770		
.STIOU	753	1284	1341#
.STOP	2366		
.SWITC	364	686#	693
.SWITS	329	714#	721
.TAKE	318	1445#	
.TQCHT	2367		
.TYPE	319	2083#	
.UMAPB	320	2158#	
.WAIT	321	1618#	
.WCS	1628	1634#	
.WCS1	1642	1648#	
.WCS2	1643	1669#	

.WES
.WSC10

1629
1634

1694#
1641#

SEQ 0332

BLDO.	158	163	166	176	183	187	188	191	195	199	200	203	206	210
	211	217	218	222	223	229								
CLOSF%	1515	2225												
CMDERR	701	734	749	777	840	846	852	892	897	1106	1139	1155	1413	1446
	1725	1733	1838	1848	1869	1879	1889	1899	1909	1919	1929	1939	1949	1959
	1969	1979	1989	1999	2008	2146	2173							
EPCAP%	596													
ERJMP	805	808	1474	1478	1516	2213	2216	2226						
FLD	159	160	161	167	168	169	170	171	172	173	174	184	185	189
	196	197	201	207	208	212	213	214	215	219	220	224	225	226
	227	357	379	382	440	443	447	449	452	456	459	475	478	481
	484	490	491	492	663	668	679	686	690	707	714	718	728	730
	743	745	759	778	798	803	819	831	854	856	883	899	901	1088
	1103	1136	1145	1271	1273	1408	1410	1450	1467	1476	1489	1548	1558	1560
	1587	1589	1618	1620	1634	1648	1669	1694	1720	1722	1832	1834	1842	1844
	1863	1865	1873	1875	1883	1885	1893	1895	1903	1905	1913	1915	1923	1926
	1933	1935	1943	1945	1953	1955	1963	1965	1973	1975	1983	1985	1993	1995
	2003	2005	2015	2017	2045	2083	2085	2140	2142	2166	2169	2182	2189	2251
	2253	2367												
FLDDB.	357	379	382	440	443	447	449	452	456	459	475	478	481	484
	490	491	492	663	668	679	686	690	707	714	718	728	730	743
	745	759	778	798	819	831	854	856	883	899	901	1088	1103	1136
	1145	1271	1273	1408	1410	1450	1467	1489	1548	1558	1560	1587	1589	1618
	1620	1634	1648	1669	1694	1720	1722	1832	1834	1842	1844	1863	1865	1873
	1875	1883	1885	1893	1895	1903	1905	1913	1915	1923	1926	1933	1935	1943
	1945	1953	1955	1963	1965	1973	1975	1983	1985	1993	1995	2003	2005	2015
	2017	2045	2083	2085	2140	2142	2166	2169	2182	2189	2251	2253		
GET%	2215													
GETAB%	631													
GETTAB	545	625												
GLOB	110													
GTJFN%	2212													
GTSTS%	1528													
GUIDE	728	743	831	883	1088	1271	1408	1558	1587	1618	1720	1832	1842	1863
	1873	1883	1893	1903	1913	1923	1933	1943	1953	1963	1973	1983	1993	2003
	2015	2083	2140	2166	2182	2189	2251							
INCHRS	2537													
ITEXT	638	2367												
JFNS%	804	1473												
JUMPF	110	957	1000	1497	2315	2425								
JUMPT	110	951	954	1008	2450									
KEYTAB	296	297	298	299	300	301	302	303	304	305	306	307	308	309
	310	311	312	313	314	315	316	317	318	319	320	321	327	328
	329	330	331	337	338	339	340	341	342	343	344	345	346	347
	348	349	350	351	352	363	364	370	371	372	373	385	391	392
	394	395	396	397	398	400	401	402	403	404	406	407	408	409
	410	411	412	413	414	415	417	419	426	427	428	429	430	431
	432	433	463	467	769	770	771	1568	1569	1570	1598	1599	1628	1629
	1642	1643	1661	1662	1663	1664	1678	1679	1680	1681	1682	1683	1684	1685
	1686	1687	1688	1689	1703	1704	1705	1706	1707	1708	1709	1710	1711	1712
	1713	1714	1715	2026	2027	2054	2055	2112	2113	2114	2115	2115	2117	2118
	2119	2120	2121	2122	2123	2124	2125	2126	2127	2128	2129	2130	2131	2132
	2133	2134	2262	2263										

SEQ 0333

LOCK	1252														SEQ 0334
LSTOF.	110	162	175	186	190	198	202	209	216	221	228	583	594	604	
	607	610	640	701	734	749	777	840	846	852	872	892	897	992	
	1074	1106	1139	1155	1178	1254	1265	1298	1304	1307	1314	1332	1333	1336	
	1342	1345	1355	1359	1363	1375	1382	1384	1392	1413	1428	1446	1503	1508	
	1725	1733	1738	1748	1756	1773	1785	1795	1807	1814	1821	1828	1838	1848	
	1869	1879	1889	1899	1909	1919	1929	1939	1949	1959	1969	1979	1989	1999	
	2008	2097	2102	2146	2154	2173	2231	2246	2272	2279	2294	2319	2320	2323	
	2330	2336	2364	2367	2370	2378	2382	2384	2398	2417	2442	2444	2457	2460	
	2466	2485	2490	2496	2499	2504	2523	2553	2571	2581	2587	2594			
LSTON.	163	176	187	191	199	203	210	217	222	229	1229	1342	1738	1748	
	1756	1773	1785	1795	1807	1814	1821	1828	2272	2294	2330	2336	2343	2364	
	2367	2398	2417	2437	2523										
MERGE.	2237														
MOVX	607	788	802	1457	1476	2210									
NAME	105														
OPENF%	1477														
OUTSTR	1542														
PBIN%	2532														
PG2ADR	568														
PIINI.	602														
PISYS.	605	608													
PJRST	110	682	702	710	993	1276	1315	1336							
POS	159	160	161	167	168	169	170	171	172	173	174	184	185	189	
	196	197	201	207	208	212	213	214	215	219	220	224	225	226	
	227	357	379	382	440	443	447	449	452	456	459	475	478	481	
	484	490	491	492	663	668	679	686	690	707	714	718	728	730	
	743	745	759	778	798	803	819	831	854	856	883	899	901	1088	
	1103	1136	1145	1271	1273	1408	1410	1450	1467	1476	1489	1548	1558	1560	
	1587	1589	1618	1620	1634	1648	1669	1694	1720	1722	1832	1834	1842	1844	
	1863	1865	1873	1875	1883	1885	1893	1895	1903	1905	1913	1915	1923	1926	
	1933	1935	1943	1945	1953	1955	1963	1965	1973	1975	1983	1985	1993	1995	
	2003	2005	2015	2017	2045	2083	2085	2140	2142	2166	2169	2182	2189	2251	
	2253	2367													
PROLOG	109														
RLJFN%	807														
RPCAP%	591														
SET0.	159	160	161	167	168	169	170	171	172	173	174	184	185	189	
	196	197	201	207	208	212	213	214	215	219	220	224	225	226	
	227														
SEVEC%	2219														
SIBEX	2529														
SKIFF	2382														
SKIPT	2380	2453													
STORE	814	816	818	1484	1485	1488									
TXNE	547	2427													
TXNN	1529														
UNLOK.	1263														
\$BUILD	158	166	183	188	195	200	206	211	218	223					
\$CALL	543	560	567	574	579	582	593	603	606	609	614	664	669	671	
	680	687	691	693	701	708	715	719	721	729	731	733	734	735	
	744	746	748	749	750	760	762	777	779	799	800	820	823	832	
	835	840	844	846	852	855	858	865	868	871	884	887	892	897	

	900	903	910	922	923	931	932	934	950	953	956	963	974	985
	991	999	1007	1066	1073	1089	1094	1104	1106	1112	1123	1128	1137	1139
	1146	1155	1158	1159	1164	1177	1200	1216	1221	1229	1231	1240	1253	1264
	1272	1274	1280	1281	1282	1283	1284	1285	1286	1291	1297	1302	1303	1305
	1306	1313	1319	1320	1325	1331	1332	1335	1340	1344	1349	1354	1358	1362
	1367	1368	1373	1374	1379	1381	1383	1388	1389	1391	1396	1397	1402	1403
	1409	1411	1413	1415	1427	1432	1433	1438	1439	1446	1451	1468	1469	1490
	1493	1496	1502	1507	1518	1543	1549	1550	1552	1559	1561	1575	1576	1581
	1582	1588	1590	1592	1605	1606	1611	1612	1619	1621	1635	1649	1652	1653
	1670	1673	1674	1695	1698	1699	1721	1723	1725	1727	1728	1733	1738	1739
	1740	1741	1742	1748	1749	1750	1756	1757	1758	1759	1760	1761	1762	1763
	1764	1765	1773	1774	1775	1776	1777	1778	1779	1785	1786	1787	1788	1789
	1795	1796	1797	1798	1800	1801	1807	1808	1814	1815	1821	1822	1828	1829
	1833	1835	1838	1843	1845	1848	1864	1866	1869	1874	1876	1879	1884	1886
	1889	1894	1896	1899	1904	1906	1909	1914	1916	1919	1924	1925	1929	1934
	1936	1939	1944	1946	1949	1954	1956	1959	1964	1966	1969	1974	1976	1979
	1984	1986	1989	1994	1996	1999	2004	2008	2016	2018	2020	2032	2033	2038
	2039	2046	2048	2060	2065	2070	2071	2084	2086	2088	2096	2101	2141	2143
	2146	2148	2151	2153	2158	2159	2167	2170	2172	2173	2174	2177	2183	2184
	2190	2191	2197	2230	2245	2252	2254	2256	2278	2314	2318	2319	2322	2342
	2348	2355	2356	2358	2366	2367	2369	2377	2378	2381	2383	2424	2436	2441
	2443	2448	2449	2452	2455	2456	2459	2465	2481	2484	2489	2495	2498	2503
	2508	2512	2515	2552	2570	2580	2586	2587	2593					
\$EOB	162	175	186	190	198	202	209	216	221	228				
\$ETAB	296#	322	327#	332	337#	353	363#	365	370#	375	385#	386	391#	421
	426#	434	463#	464	467#	468	769#	772	1568#	1571	1598#	1600	1628#	1630
	1642#	1644	1661#	1665	1678#	1690	1703#	1716	2026#	2028	2054#	2056	2112#	2135
\$RET	2262#	2264												
	110#	110	628	634	696	724	737	764	861	875	906	914	1030	1051
	1074	1178	1192	1217	1222	1250	1255	1261	1266	1287	1298	1321	1345	1363
	1369	1375	1384	1392	1398	1404	1428	1434	1440	1503	1577	1583	1607	1613
	1654	1675	1700	1729	1743	1751	1766	1780	1790	1802	1809	1816	1823	1830
	1840	1850	1871	1881	1891	1901	1911	1921	1931	1941	1951	1961	1971	1981
	1991	2001	2010	2034	2040	2061	2066	2072	2107	2154	2161	2178	2231	2285
	2307	2320	2323	2328	2359	2375	2384	2410	2412	2432	2444	2514	2531	2538
	2553	2571	2581	2594										
\$RETF	110#													
\$RETIF	110#													
\$RETIT	110#													
\$RETT	110#													
\$SAVE	1229	1341	1737	1747	1755	1772	1784	1794	1806	1813	1820	1827	2271	2293
	2329	2335	2342	2363	2397	2416	2436	2522						
\$SET	159	160	161	167	168	169	170	171	172	173	174	184	185	189
	196	197	201	207	208	212	213	214	215	219	220	224	225	226
	227													
\$STAB	295	326	336	362	369	384	390	425	462	466	768	1567	1597	1627
	1641	1660	1677	1702	2025	2053	2111	2261						
\$STOP	2366													
\$TEXT	580	593	603	606	609	701	734	749	777	840	846	852	871	892
	897	991	1073	1106	1139	1155	1177	1253	1264	1297	1303	1306	1313	1331
	1332	1335	1344	1353	1358	1362	1374	1381	1383	1390	1413	1425	1446	1502
	1507	1725	1733	1838	1848	1869	1879	1889	1899	1909	1919	1929	1939	1949
	1959	1969	1979	1989	1999	2008	2095	2100	2146	2153	2173	2230	2245	2278

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55

```
;Z:<GSCOTT.DFCIB>DFCIBM.MAC.646 16-Aug-85 13:46:38, Edit by GSCOTT
;In variable length tests, output errors including packet size
;Z:<GSCOTT.DFCIB>DFCIBM.MAC.637 16-Aug-85 10:14:43, Edit by GSCOTT
;Remove extern of SWPALL and SWHALT, since not used
;Z:<GSCOTT.DFCIB>DFCIBM.MAC.636 15-Aug-85 21:00:44, Edit by GSCOTT
;Only output those strange messages in TST58/59 if trace mode.
;Z:<GSCOTT.DFCIB>DFCIBM.MAC.635 14-Aug-85 18:30:44, Edit by GSCOTT
;Remove VMSERR, HSCERR, HSCONL
;Z:<GSCOTT.DFCIB>DFCIBM.MAC.634 14-Aug-85 18:07:03, Edit by GSCOTT
;Intern RNVNUM
;Z:<GSCOTT.DFCIB>DFCIBM.MAC.633 14-Aug-85 18:00:34, Edit by GSCOTT
;Replace RSPER2, STAER2, CNTER2, RSPER3, STAER3, CNTER3, RNVERR
;Z:<GSCOTT.DFCIB>DFCIBM.MAC.631 14-Aug-85 17:49:31, Edit by GSCOTT
;Replace RSPERO, STAERO, start removing RNVERR, RSPER1, STAER1, CNTER1
;Z:<GSCOTT.DFCIB>DFCIBM.MAC.630 14-Aug-85 17:31:24, Edit by GSCOTT
;Replace RSPERR, STAERR, CNTER with proper ERROR1 calls
;Z:<GSCOTT.DFCIB>DFCIBM.MAC.629 14-Aug-85 17:04:37, Edit by GSCOTT
;Clean up ERROR1 calls that don't specify an actual/expected word
;Z:<GSCOTT.DFCIB>DFCIBM.MAC.625 14-Aug-85 16:51:30, Edit by GSCOTT
;Remove unused DMAVL subroutine.
;Z:<GSCOTT.DFCIB>DFCIBM.MAC.622 14-Aug-85 16:44:14, Edit by GSCOTT
;Move variable data to end of module
;Z:<GSCOTT.DFCIB>DFCIBM.MAC.621 14-Aug-85 16:44:01, Edit by GSCOTT
;Replace DMAER1, DATER6, DATER7, DATER8, DATERB
;Z:<GSCOTT.DFCIB>DFCIBM.MAC.620 14-Aug-85 16:22:58, Edit by GSCOTT
;Replce M58ER1, M58ER2, M59ER1, M59ER2 with proper ERROR1 calls
;Z:<GSCOTT.DFCIB>DFCIBM.MAC.617 14-Aug-85 15:47:41, Edit by GSCOTT
;Replace MPER52, MPER53, MPER54, M56ER1, M57ER1 with proper ERROR1 calls
;Z:<GSCOTT.DFCIB>DFCIBM.MAC.616 14-Aug-85 15:38:30, Edit by GSCOTT
;Replace UNMPER with proper ERROR1 calls.
;Z:<GSCOTT.DFCIB>DFCIBM.MAC.615 14-Aug-85 15:15:44, Edit by GSCOTT
;Delete extra copies of RDSTIM and OTIME here.
;Z:<GSCOTT.DFCIB>DFCIBM.MAC.614 14-Aug-85 15:14:18, Edit by GSCOTT
;Remov CONER6, CONER7, CONER8, CONER9
;Z:<GSCOTT.DFCIB>DFCIBM.MAC.612 14-Aug-85 14:11:41, Edit by GSCOTT
;Change error messages to use proper ERROR1 calls: CNERR6, CNERR8, CNERR7
;Z:<GSCOTT.DFCIB>DFCIBM.MAC.609 13-Aug-85 13:51:32, Edit by GSCOTT
;Change test messages for TRNODE slightly
;Z:<GSCOTT.DFCIB>DFCIBM.MAC.608 13-Aug-85 13:20:38, Edit by GSCOTT
;Remove call to counter test at end of exerciser script
;Z:<GSCOTT.DFCIB>DFCIBM.MAC.602 12-Aug-85 22:28:24, Edit by GSCOTT
;Move TSTTAB and SCRTAB to DFNIBP
;Z:<GSCOTT.DFCIB>DFCIBM.MAC.599 12-Aug-85 17:26:39, Edit by GSCOTT
;Move ERRHAN to DFCIBP
;Z:<GSCOTT.DFCIB>DFCIBM.MAC.597 12-Aug-85 12:22:28, Edit by GSCOTT
;Remove unreferenced externs EVENTB, GEVNT, GETFN, GETNN, LNODE, LOFSET, NAMES,
;RDAT1, ROFSET, RSCSE, SCSERR, DMADRS, CTPPKT, BUFRNM, BUFLNM,
;error message CNTER0, M51ER3, cell MAXBUF
;Z:<GSCOTT.DFCIB>DFCIBM.MAC.595 9-Aug-85 14:49:41, Edit by GSCOTT
;TST55 didn't compare bytes right AT ALL. Fix it.
;Z:<GSCOTT.DFCIB>DFCIBM.MAC.589 9-Aug-85 14:19:47, Edit by GSCOTT
;Create NNNBYL to output buffer length and which byte.
;Z:<GSCOTT.DFCIB>DFCIBM.MAC.579 9-Aug-85 12:21:58, Edit by GSCOTT
;Replace DATER1, DATER2, DATER3, DATER4, DATER5 with ERROR1s, create NNNBYT.
;Z:<GSCOTT.DFCIB>DFCIBM.MAC.571 9-Aug-85 10:33:55, Edit by GSCOTT
```


56 :Expected count from buffer unmap command is 1. Fix all TST5x tests.
 57 :Z:<GSCOTT.DFCIB>DFCIBM.MAC.564 9-Aug-85 09:57:49, Edit by GSCOTT
 58 :Remove NOSYM
 59 :Z:<GSCOTT.DFCIB>DFCIBM.MAC.545 8-Aug-85 23:17:11, Edit by GSCOTT
 60 :At T50C didn't set up BUFTYP as the expected status from map buffer.
 61 :Z:<GSCOTT.DFCIB>DFCIBM.MAC.539 8-Aug-85 21:59:55, Edit by GSCOTT
 62 :Delete ITCNT1, ITCNT externs.
 63 :Z:<GSCOTT.DFCIB>DFCIBM.MAC.538 8-Aug-85 19:20:19, Edit by GSCOTT
 64 :Missed some calls to ERM14A
 65 :Z:<GSCOTT.DFCIB>DFCIBM.MAC.537 8-Aug-85 18:29:09, Edit by GSCOTT
 66 :Use correct ERROR1 macro inplace of ERM14A and ERM57A
 67 :Z:<GSCOTT.DFCIB>DFCIBM.MAC.535 8-Aug-85 18:05:46, Edit by GSCOTT
 68 :Remove unreferenced CNERR4, CNERR5, CNERR9, CONER4, CONER5, T2ERR1, T2ERR2
 69 :Replace CONER1, CONER2, CONER3 with ERROR1 macros.
 70 :Z:<GSCOTT.DFCIB>DFCIBM.MAC.528 8-Aug-85 17:50:17, Edit by GSCOTT
 71 :Remove CNERR1, CNERR2, CNERR3 in favor of better use of ERROR1 macro.
 72 :Z:<GSCOTT.DFCIB>DFCIBM.MAC.525 8-Aug-85 17:34:54, Edit by GSCOTT
 73 :Fix use of RCVRM in TST50.
 74 :Z:<GSCOTT.DFCIB>DFCIBM.MAC.514 8-Aug-85 15:47:15, Edit by GSCOTT
 75 :Move SCRTAB here from DFCIBP, put just tests in test table, scripts in SCRTAB.
 76 :Z:<GSCOTT.DFCIB>DFCIBM.MAC.513 8-Aug-85 15:38:47, Edit by GSCOTT
 77 :Indent output of TST80 (read counters) to make it easier to read.
 78 :Z:<GSCOTT.DFCIB>DFCIBM.MAC.507 8-Aug-85 15:25:47, Edit by GSCOTT
 79 :Replace CONER5 calls with proper ERROR1 macro
 80 :Z:<GSCOTT.DFCIB>DFCIBM.MAC.502 8-Aug-85 15:15:39, Edit by GSCOTT
 81 :Output decimal numbers in for ERROR1 call if first arg negative.
 82 :Z:<GSCOTT.DFCIB>DFCIBM.MAC.500 8-Aug-85 15:02:46, Edit by GSCOTT
 83 :Output better message if trace mode.
 84 :Z:<GSCOTT.DFCIB>DFCIBM.MAC.498 8-Aug-85 14:36:00, Edit by GSCOTT
 85 :Extern PCRLF, reorder externs by module.
 86 :Z:<GSCOTT.DFCIB>DFCIBM.MAC.496 8-Aug-85 14:25:54, Edit by GSCOTT
 87 :Fix ERRHAN to output cleaner format with blank lines between errors
 88 :Z:<GSCOTT.DFCIB>DFCIBM.MAC.487 7-Aug-85 18:31:19, Edit by GSCOTT
 89 :Remove DATSTA, DATSTB and replace with proper constants.
 90 :Z:<GSCOTT.DFCIB>DFCIBM.MAC.485 7-Aug-85 18:25:36, Edit by GSCOTT
 91 :Remove IMGDA1 and IMGDA2 and assorted crufty code, use MOVEI instead.
 92 :Z:<GSCOTT.DFCIB>DFCIBM.MAC.478 7-Aug-85 17:26:32, Edit by GSCOTT
 93 :Code at TST80H doesn't work, so repeat 0 it.
 94 :Z:<GSCOTT.DFCIB>DFCIBM.MAC.477 7-Aug-85 17:26:09, Edit by GSCOTT
 95 :REVRSN should print numbers in decimal.
 96 :Z:<GSCOTT.DFCIB>DFCIBM.MAC.470 7-Aug-85 17:11:12, Edit by GSCOTT
 97 :Fix ERRHAN to output daytime at error.
 98 :Z:<GSCOTT.DFCIB>DFCIBM.MAC.465 7-Aug-85 14:58:13, Edit by GSCOTT
 99 :Use MAPB8A (page from GLXLIB) for address of MAPB8.
 100 :Z:<GSCOTT.DFCIB>DFCIBM.MAC.464 7-Aug-85 13:02:15, Edit by GSCOTT
 101 :Remove GETD/PUTD, SETDP
 102 :Z:<GSCOTT.DFCIB>DFCIBM.MAC.451 7-Aug-85 01:08:03, Edit by GSCOTT
 103 :Add ERRHAN subroutine.
 104 :Z:<GSCOTT.DFCIB>DFCIBM.MAC.442 6-Aug-85 23:30:06, Edit by GSCOTT
 105 :Remove references to ACO.
 106 :Z:<GSCOTT.DFCIB>DFCIBM.MAC.432 6-Aug-85 20:42:38, Edit by GSCOTT
 107 :Remove tests that are commented out to save myself work.
 108 :Z:<GSCOTT.DFCIB>DFCIBM.MAC.431 6-Aug-85 20:41:17, Edit by GSCOTT
 109 :The big change: ACS, title statements, etc. Remove UTLTAB.
 110 :Z:<GSCOTT.DFCIB>DFCIBM.MAC.425 6-Aug-85 17:19:44, Edit by GSCOTT

```
111      ;Remove PGNAME and PROMPT, etc.  
112      ;Z:<GSCOTT.DFCIB>DFCIBM.MAC.421 6-Aug-85 17:05:33, Edit by GSCOTT  
113      ;Remove STAT and STATB from extern - not used  
114      ;Z:<GSCOTT.DFCIB>DFCIBM.MAC.419 5-Aug-85 14:09:12, Edit by GSCOTT  
115      ;WAITX argument to S1  
116      ;Z:<GSCOTT.DFCIB>DFCIBM.MAC.418 5-Aug-85 13:38:54, Edit by GSCOTT  
117      ;RCON arguments in S1,S2.  
118      ;Z:<GSCOTT.DFCIB>DFCIBM.MAC.416 5-Aug-85 12:04:05, Edit by GSCOTT  
119      ;Start GLXLIB conversion: move SETUP code to DFCIBP.  
120  
121      ; 5/8/85 DELETED REFERENCE TO WAITE,WAITC IN EXTERNAL  
122      ; 7-31-85 ADDED REFERENCE TO T10DST IN TST51,TST59.
```


SUBTTL Definitions

SEARCH DFCIBT

SALL

.DIREC FLBLST

NAME (DFCIBM,\DECVER,\VEDIT,CTP Tests 01-80)

SEARCH GLXMAC,MONSYM,UUOSYM,MACSYM

PROLOG (DFCIBM)

;Externs found in DFCIB1

EXTERN DOCONX,MINMS,RSPMXM,DOCON,EWAIT,CWAIT,UMPBUF,WAITX,SCSERS

EXTERN LNODEN,MAPBL,RCON,RCONB,CONNB,MPBUF

;Externs found in DFCIB2

EXTERN DODIS,READM,SNDSMSG,DMARD,REQUEM,OPCODE,BLDCTP,RETBUF

EXTERN BUFTYP,GENFUN,GNCNST,BUFLEN,GENLEN,NUMREF

EXTERN DELAY,RCOUNT,OTHNOD,MOVTYP,PKTSIZ,PKTMLT

EXTERN READD,SNDDAT,REQUED

;Externs found in DFCIB3

EXTERN SDAT20,SDATVM,RDAT20,RDATVM

;Externs found in DFCIBY

EXTERN TST99

;Externs found in DFCIBP

EXTERN SWTRAC,TSTNAM,TSTDSC,PCRLF,CHKST,CHKEV,MAPBBA

EXTERN NODEN,DESNA,SELNA,ERRHAN

;Interns

INTERN TST01,TST02

INTERN TST10,TST11,TST12,TST13,TST14

INTERN TST30,TST31,TST32,TST33

INTERN TST40

INTERN TST50,TST51,TST52,TST53,TST54,TST55,TST56,TST57,TST58,TST59

INTERN TST80

INTERN TALL,BALL,MALL,DALL,BUFALL,CNTALL,XRCSE,DEF

INTERN IMAGDT,IMGLN,ITERCT,HSCDST,T20DST,VMDSST,T10DST,FLGFLG,RNVNUM

;INTERN'S ADDED FOR TST99 (DFCIBY)

INTERN TEVPND,CONACC,CONOPN,RCVRSP,RCVACT

INTERN SNDCMP,XMTREQ,RNVERF

INTERN CNTFLG,EXPACT,EXPSTA,RCVRM,EXPRSP,RCVSTA,TSTACT

SUBTTL Scripts

;Here are the scripts

;All-tests

174
175
176
177
178
179
180 000000' 260 17 0 00 000006'
181 000001' 260 17 0 00 000011'
182 000002' 260 17 0 00 000017'
183 000003' 260 17 0 00 000025'
184 000004' 260 17 0 00 000040'
185 000005' 263 17 0 00 000000

TALL: \$CALL BALL
\$CALL MALL
\$CALL DALL
\$CALL BUFALL
\$CALL CNTALL
\$RET

;Basic tests
;Message tests
;Datagram tests
;Buffer tests
;Counter test

;Basic tests

186
187
188
189 000006' 260 17 0 00 000053'
190 000007' 260 17 0 00 000077'
191 000010' 263 17 0 00 000000

BALL: \$CALL TST01
\$CALL TST02
\$RET

;Run Test-01
;Run Test-02

;Message tests

192
193
194
195 000011' 260 17 0 00 000150'
196 000012' 260 17 0 00 000232'
197 000013' 260 17 0 00 000545'
198 000014' 260 17 0 00 001530'
199 000015' 260 17 0 00 002041'
200 000016' 263 17 0 00 000000

MALL: \$CALL TST10
\$CALL TST11
\$CALL TST12
\$CALL TST13
\$CALL TST14
\$RET

;Run Test-10
;Run Test-11
;Run Test-12
;Run Test-13
;Run Test-14

;Data tests

201
202
203
204 000017' 260 17 0 00 002465'
205 000020' 260 17 0 00 002561'
206 000021' 260 17 0 00 003156'
207 000022' 260 17 0 00 004452'
208 000023' 260 17 0 00 005061'
209 000024' 263 17 0 00 000000

DALL: \$CALL TST30
\$CALL TST31
\$CALL TST32
\$CALL TST33
\$CALL TST40
\$RET

;Run Test-30
;Run Test-31
;Run Test-32
;Run Test-33
;Run Test-40

210									
211									
212									
213	000025'	260	17	0	00	005242'	BUFALL: \$CALL	TST50	;Run Test-50
214	000026'	260	17	0	00	005360'	\$CALL	TST51	;Run Test-51
215	000027'	260	17	0	00	006156'	\$CALL	TST52	;Run Test-52
216	000030'	260	17	0	00	006406'	\$CALL	TST53	;Run Test-53
217	000031'	260	17	0	00	006636'	\$CALL	TST54	;Run Test-54
218	000032'	260	17	0	00	007071'	\$CALL	TST55	;Run Test-55
219	000033'	260	17	0	00	007341'	\$CALL	TST56	;Run Test-56
220	000034'	260	17	0	00	007574'	\$CALL	TST57	;Run Test-57
221	000035'	260	17	0	00	010030'	\$CALL	TST58	;Run Test-58
222	000036'	260	17	0	00	010364'	\$CALL	TST59	;Run Test-59
223	000037'	263	17	0	00	000000	\$RET		;Return to parser
224									
225									
226									
227	000040'	260	17	0	00	010701'	CNTALL: \$CALL	TST80	;Counter test
228	000041'	263	17	0	00	000000	\$RET		
229									
230									
231									
232	000042'	260	17	0	00	000000*	XRCSE: \$CALL	TST99	;Exerciser
233	000043'	263	17	0	00	000000	\$RET		
234									
235									
236									
237	000044'	332	00	0	00	000000*	DEF: SKIPE	SWTRAC	;Trace mode?
238							\$TEXT	(,<Running Default Script	
239	000045'	260	17	0	00	000000*	Deselecting all	nodes and selecting all	known nodes>)
240	000047'	260	17	0	00	000000*	\$CALL	DESNA	;Deselect all nodes
241	000050'	260	17	0	00	000000*	\$CALL	SELNA	;Select all nodes monitor knows about
242	000051'	260	17	0	00	000000'	\$CALL	TALL	;Go run all tests
243	000052'	263	17	0	00	000000	\$RET		;Return to parser

244
 245
 246
 247
 248
 249
 250 000053' 201 01 0 00 011535'
 251 000054' 201 02 0 00 011537'
 252 000055' 260 17 0 00 011372'
 253
 254
 255
 256 000056' 260 17 0 00 011134'
 257 000057' 254 00 0 00 000066'
 258
 259 000060' 260 17 0 00 011562'
 260 000061' 254 00 0 00 000076'
 261
 262 000062' 260 17 0 00 011607'
 263 000063' 254 00 0 00 000076'
 264 000064' 260 17 0 00 011630'
 265 000065' 254 00 0 00 000076'
 266
 267 000066' 201 01 0 00 000764
 268 000067' 260 17 0 00 000000*
 269
 270
 271
 272 000070' 260 17 0 00 000000*
 273
 274
 275
 276 000071' 260 17 0 00 011306'
 277 000072' 254 00 0 00 000074'
 278 000073' 260 17 0 00 011650'
 279
 280
 281
 282 000074' 201 01 0 00 000764
 283 000075' 260 17 0 00 000067*
 284
 285 000076' 263 17 0 00 000000

SUBTTL TST01 Connect/Disconnect Test

 ;* TST01 Connect then Disconnect to selected node.
 ;*****

TST01: MOVEI S1,[ASCIZ/TST01/] ;Load test ASCIZ
 MOVEI S2,[ASCIZ\Connect/Disconnect Test\]
 \$CALL TRNODE ;Print node number if trace switch set

;Connect with selected node and verify connection=open.

\$CALL CONNCT ;Connect to node
 JRST TST01A ; JUMP IF CONNECTION OK.
 ERROR1 (,,,Timed out waiting for event pending flag during connection attempt,,NNN)

JRST TST01X
 ERROR1 (,,,Event pending was not connection accepted during connection attempt,,NNN

)

JRST TST01X
 ERROR1 (,,,Connection not open after connection accepted,,NNN)
 JRST TST01X

TST01A: MOVEI S1,^D500 ; WAIT FOR 500 MILLESECONDS.
 \$CALL WAITX

;Disconnect with selected node

\$CALL DODIS ;Disconnect with the selected node

;Wait for connection closed

\$CALL CONCLD
 JRST TST01B
 ERROR1 (,,,Connection closed not in disconnect status,,NNN)

;Wait

TST01B: MOVEI S1,^D500
 \$CALL WAITX

TST01X: \$RET

286
 287
 288
 289
 290
 291
 292 000077' 201 01 0 00 011657'
 293 000100' 201 02 0 00 011661'
 294 000101' 260 17 0 00 011372'
 295
 296
 297
 298 000102' 260 17 0 00 011134'
 299 000103' 254 00 0 00 000112'
 300
 301 000104' 260 17 0 00 011562'
 302 000105' 254 00 0 00 000145'
 303
 304 000106' 260 17 0 00 011607'
 305 000107' 254 00 0 00 000145'
 306 000110' 260 17 0 00 011630'
 307 000111' 254 00 0 00 000145'
 308
 309
 310
 311 000112' 402 00 0 00 000000*
 312 000113' 260 17 0 00 000000*
 313
 314
 315
 316 000114' 260 17 0 00 011111'
 317 000115' 254 00 0 00 000124'
 318
 319 000116' 260 17 0 00 011707'
 320 000117' 254 00 0 00 000145'
 321 000120' 260 17 0 00 011733'
 322 000121' 254 00 0 00 000145'
 323
 324 000122' 260 17 0 00 011760'
 325 000123' 254 00 0 00 000145'

SUBTTL TST02 - Transmit Function Set Request/Receive Response

 * TST02 - TRANSMIT FUNCTION SET REQUEST/RECEIVE RESPONSE *

TST02: MOVEI S1,[ASCIZ/TST02/] ;Point to test name
 MOVEI S2,[ASCIZ\Function Set Request/Response\]
 \$CALL TRNODE ;Print node number if trace switch set

;Connect with selected node and verify connection=open.

\$CALL CONNCT
 JRST TST02A ; JUMP IF CONNECTION OK.
 ERROR1 (,,,Timed out waiting for event pending flag during connection attempt,,NNN)

JRST TST02Y
 ERROR1 (,,,Event pending was not connection accepted during connection attempt,,NNN)

)
 JRST TST02Y
 ERROR1 (,,,Connection not open after connection accepted,,NNN)
 JRST TST02Y

;Build CTP 0 MSG

TST02A: SETZM OP CODE ;SET OP CODE = 0.
 \$CALL BLDCTP ;Build packet

;Transmit request message

\$CALL XMTREQ ;Transmit request
 JRST TST02B ;OK
 ERROR1 (,,,<Timed out waiting for event pending after transmitting function set mes
 sage >,,NNN)

JRST TST02Y
 ERROR1 (,,,<No send complete event after transmitting function set message>,,NNN)

JRST TST02Y
 ERROR1 (,,,<Timed out waiting for function set response message available flag>,,NN

N)
 JRST TST02Y


```

326
327
328
329 000124' 402 00 0 00 011430'
330 000125' 402 00 0 00 011422'
331 000126' 260 17 0 00 011043'
332 000127' 254 00 0 00 000137'
333 000130' 260 17 0 00 011777'
334 000131' 254 00 0 00 000137'
335 000132' 260 17 0 00 012016'
336 000133' 254 00 0 00 000137'
337 000134' 260 17 0 00 012035'
338 000135' 254 00 0 00 000137'
339 000136' 260 17 0 00 012056'
340
341
342
343 000137' 260 17 0 00 000000*
344
345
346
347 000140' 260 17 0 00 000070*
348
349
350
351 000141' 260 17 0 00 011306'
352 000142' 254 00 0 00 000145'
353 000143' 260 17 0 00 011650'
354 000144' 254 00 0 00 000147'
355
356 000145' 260 01 0 00 000764
357 000146' 260 17 0 00 000075*
358
359 000147' 263 17 0 00 000000

;Receive response message, test <RESPONSE=64> and <STATUS=0>
TST02B: SETZM EXPSTA
        SETZM CNTFLG
        $CALL RCVRM
        JRST TST02W
        ERROR1 (-1,EXPRSP,RCVRSP,Unexpected function set response opcode,,NNN)
        JRST TST02W
        ERROR1 (-1,EXPSTA,RCVSTA,Unexpected function set response status,,NNN)
        JRST TST02W
        ERROR1 (-1,EXPSTA,RCVSTA,Unexpected function set response count,,NNN)
        JRST TST02W
        ERROR1 (1,NUMREF,RNVNUM,Unexpected function set response reference number,,NNN)

;Requeue the buffer
TST02W: $CALL REQUEM

;Disconnect with selected node
TST02X: $CALL DODIS ;Disconnect with the selected node

;Wait for connection closed
        $CALL CONCLD
        JRST TST02Y
        ERROR1 (1,CONCLD,Connection closed not in disconnect status,,NNN)
        JRST TST02Z
TST02Y: MOVEI S1,^D500
        $CALL WAITX
TST02Z: $RET

```

SUBTTL TST10 - Basic Message Test

```

*****
* TST10 - BASIC MESSAGE TEST. THIS TEST WILL SEND A MESSAGE REQUEST OF
* DATA=0, LENGTH=0, DELAY=0, REPEAT COUNT=0.
*****

```

```

TST10:  MOVEI  S1,[ASCIZ/TST10/]      ;Load test name
        MOVEI  S2,[ASCIZ/Basic Message Test (Count=0)/]
        $CALL  TRNODE                 ;Print node number if trace switch set

```

;Connect with selected node and verify connection=open.

```

$CALL  CONNCT

```

```

JRST   TST10A                      ; JUMP IF CONNECTION OK.
ERROR1 (,,,Timed out waiting for event pending flag during connection attempt,,NNN)

```

```

JRST   TST10X
ERROR1 (,,,Event pending was not connection accepted during connection attempt,,NNN)

```

```

)
JRST   TST10X
ERROR1 (,,,Connection not open after connection accepted,,NNN)
JRST   TST10X

```

;Build and send generate message ctp packet

```

TST10A: MOVEI  S1,4
        MOVEM  S1,OPCODE              ; OPCODE = GEN MESSAGE.

```

```

        MOVEI  S1,0
        MOVEM  S1,GENFUN              ; GENFUNCT = 0/GENFILL.
        MOVEM  S1,DELAY               ; DELAY = 0.
        MOVEM  S1,GENLEN              ; GEN LENGTH = 0.
        MOVEM  S1,GNCNST              ; GENCONST = 0.
        MOVEM  S1,RCOUNT              ; REPEAT COUNT = 0.

```

```

$CALL  BLDCTP

```

```

$CALL  XMTREQ                      ;Transmit request
JRST   TST10B                      ;OK

```

```

ERROR1 (,,,<Timed out waiting for event pending after transmitting generate message
>,,NNN)

```

```

JRST   TST10W
ERROR1 (,,,<No send complete event after transmitting generate message>,,NNN)

```

```

JRST   TST10W
ERROR1 (,,,<Timed out waiting for generate message response available flag>,,NNN)
JRST   TST10W

```

```

360
361
362
363
364
365
366
367 000150' 201 01 0 00 012065'
368 000151' 201 02 0 00 012067'
369 000152' 260 17 0 00 011372'
370
371
372
373 000153' 260 17 0 00 011134'
374
375 000154' 254 00 0 00 000163'
376
377 000155' 260 17 0 00 011562'
378 000156' 254 00 0 00 000222'
379
380 000157' 260 17 0 00 011607'
381 000160' 254 00 0 00 000222'
382 000161' 260 17 0 00 011630'
383 000162' 254 00 0 00 000222'
384
385
386
387 000163' 201 01 0 00 000004
388 000164' 202 01 0 00 000112*
389
390 000165' 201 01 0 00 000000
391 000166' 202 01 0 00 000000*
392 000167' 202 01 0 00 000000*
393 000170' 202 01 0 00 000000*
394 000171' 202 01 0 00 000000*
395 000172' 202 01 0 00 000000*
396
397 000173' 260 17 0 00 000113*
398
399 000174' 260 17 0 00 011111'
400 000175' 254 00 0 00 000204'
401
402 000176' 260 17 0 00 012114'
403 000177' 254 00 0 00 000221'
404 000200' 260 17 0 00 012137'
405 000201' 254 00 0 00 000221'
406 000202' 260 17 0 00 012163'
407 000203' 254 00 0 00 000221'

```

408
 409
 410
 411 000204' 402 00 0 00 011430'
 412 000205' 476 00 0 00 011422'
 413 000206' 402 00 0 00 011423'
 414 000207' 260 17 0 00 011043'
 415
 416 000210' 254 00 0 00 000221'
 417 000211' 260 17 0 00 012203'
 418 000212' 254 00 0 00 000221'
 419 000213' 260 17 0 00 012223'
 420 000214' 254 00 0 00 000221'
 421 000215' 260 17 0 00 012243'
 422 000216' 254 00 0 00 000221'
 423
 424 000217' 260 17 0 00 012265'
 425 000220' 254 00 0 00 000221'
 426
 427
 428
 429 000221' 260 17 0 00 000137*
 430
 431
 432
 433 000222' 260 17 0 00 000140*
 434
 435
 436
 437 000223' 260 17 0 00 011306'
 438 000224' 254 00 0 00 000227'
 439 000225' 260 17 0 00 011650'
 440 000226' 254 00 0 00 000222'
 441
 442
 443
 444 000227' 201 01 0 00 000764
 445 000230' 260 17 0 00 000146*
 446
 447 000231' 263 17 0 00 000000

;Read the message response

TST10B: SETZM EXPSTA ; EXPECTED STATUS=0.
 SETOM CNTFLG ; TEST COUNT FIELD.
 SETZM EXPACT ; EXPECTED ACTUAL CNT=0.
 \$CALL RCVRM
 JRST TST10W
 ERROR1 (-1,EXPRSP,RCVRSP,Unexpected generate message response opcode,,NNN)
 JRST TST10W
 ERROR1 (-1,EXPSTA,RCVSTA,Unexpected generate message response status,,NNN)
 JRST TST10W
 ERROR1 (-1,EXPACT,RCVACT,Unexpected generate message response count,,NNN)
 JRST TST10W
 ERROR1 (1,NUMREF,RNVNUM,Unexpected generate message response reference number,,NNN)
 JRST TST10W

;Requeue the buffer

TST10W: \$CALL REQUEM

;Disconnect with selected node

TST10X: \$CALL DODIS ;Disconnect with the selected node

;Wait for connection closed

\$CALL CONCLO
 JRST TST10Y
 ERROR1 (,,Connection closed not in disconnect status,,NNN)
 JRST TST10X

;Wait

TST10Y: MOVEI S1,^D500
 \$CALL WAITX

TST10Z: \$RET

448
 449
 450
 451
 452
 453
 454
 455 000232' 201 01 0 00 012274'
 456 000233' 201 02 0 00 012276'
 457 000234' 260 17 0 00 011372'
 458
 459
 460
 461 000235' 260 17 0 00 011134'
 462 000236' 254 00 0 00 000245'
 463
 464 000237' 260 17 0 00 011562'
 465 000240' 254 00 0 00 000542'
 466
 467 000241' 260 17 0 00 011607'
 468 000242' 254 00 0 00 000542'
 469 000243' 260 17 0 00 011630'
 470 000244' 254 00 0 00 000542'
 471
 472
 473
 474 000245' 201 01 0 00 000004
 475 000246' 202 01 0 00 000164*
 476 000247' 202 01 0 00 000170*
 477 000250' 202 01 0 00 011510'
 478
 479 000251' 201 01 0 00 000000
 480 000252' 202 01 0 00 000167*
 481 000253' 202 01 0 00 000172*
 482 000254' 202 01 0 00 000166*
 483
 484 000255' 201 01 0 00 000377
 485 000256' 202 01 0 00 000171*
 486
 487 000257' 260 17 0 00 000173*

SUBTTL TST11 Message Test - 4 bytes each of 4 data types

 ; * TST11 - MESSAGE TEST. THIS TEST WILL SEND A MESSAGE REQUEST OF
 ; DATA=377, LENGTH=4, DELAY=0, REPEAT COUNT=0.
 ; *****

TST11: MOVEI S1,[ASCIIZ/TST11/] ;Load test name
 MOVEI S2,[ASCIIZ/Message Test (Count=4, Data=377)/]
 \$CALL TRNODE ;Print node number if trace switch set

;Connect with selected node and verify connection=open.

\$CALL CONNCT ;Connect with the selected node
 JRST TST11B
 ERROR1 (,,,Timed out waiting for event pending flag during connection attempt,,NNN)

JRST TST11Y
 ERROR1 (,,,Event pending was not connection accepted during connection attempt,,NNN)

)
 JRST TST11Y
 ERROR1 (,,,Connection not open after connection accepted,,NNN)
 JRST TST11Y

;Build and send generate message ctp packet

TST11B: MOVEI S1,4
 MOVEM S1,OPCODE ; OP CODE = GEN MESSAGE.
 MOVEM S1,GENLEN ; GEN LENGTH=4.
 MOVEM S1,ITERCT

 MOVEI S1,0
 MOVEM S1,DELAY ; DELAY = 0.
 MOVEM S1,RCOUNT ; REPEAT COUNT = 0.
 MOVEM S1,GENFUN ; GENFUNCT =0/GENFILL

 MOVEI S1,377
 MOVEM S1,GNCNST ; GENCONST=377.

 \$CALL BLDCTP

488
 489
 490
 491 000260' 260 17 0 00 011111'
 492 000261' 254 00 0 00 000270'
 493
 494 000262' 260 17 0 00 012114'
 495 000263' 254 00 0 00 000535'
 496 000264' 260 17 0 00 012137'
 497 000265' 254 00 0 00 000535'
 498 000266' 260 17 0 00 012163'
 499 000267' 254 00 0 00 000535'
 500
 501
 502
 503 000270' 402 00 0 00 011430'
 504 000271' 402 00 0 00 011423'
 505 000272' 476 00 0 00 011422'
 506 000273' 260 17 0 00 011043'
 507
 508 000274' 254 00 0 00 000305'
 509 000275' 260 17 0 00 012203'
 510 000276' 254 00 0 00 000321'
 511 000277' 260 17 0 00 012223'
 512 000300' 254 00 0 00 000321'
 513 000301' 260 17 0 00 012243'
 514 000302' 254 00 0 00 000321'
 515
 516 000303' 260 17 0 00 012305'
 517 000304' 254 00 0 00 000321'
 518
 519
 520
 521 000305' 402 00 0 00 000003
 522 000306' 200 01 0 00 000000*
 523 000307' 200 02 0 00 012314'
 524
 525 000310' 134 06 0 00 000002
 526 000311' 316 06 0 00 000256*
 527 000312' 254 00 0 00 000316'
 528 000313' 200 04 0 00 000311*
 529 000314' 260 17 0 00 012321'
 530 000315' 254 00 0 00 000321'
 531
 532
 533
 534 000316' 271 03 0 00 000001
 535 000317' 312 03 0 00 011510'
 536 000320' 254 00 0 00 000310'
 537
 538
 539
 540 000321' 260 17 0 00 000221*

;TRANSMIT REQUEST MESSAGE

```

$CALL XMTREQ ;Transmit request
JRST TST11C ;OK
ERROR1 (,,, <Timed out waiting for event pending after transmitting generate message
>,,NNN)
JRST TST11X
ERROR1 (,,, <No send complete event after transmitting generate message>,,NNN)
JRST TST11X
ERROR1 (,,, <Timed out waiting for generate message response available flag>,,NNN)
JRST TST11X

```

;RECEIVE RESPONSE MESSAGE

```

TST11C: SETZM EXPSTA
SETZM EXPACT
SETOM CNTFLG
$CALL RCVRM

JRST TST11D
ERROR1 (-1,EXPRSP,RCVRSP,Unexpected generate message response opcode,,NNN)
JRST TST11G
ERROR1 (-1,EXPSTA,RCVSTA,Unexpected generate message response status,,NNN)
JRST TST11G
ERROR1 (-1,EXPACT,RCVACT,Unexpected generate message response count,,NNN)
JRST TST11G
ERROR1 (1,NUMREF,RNVNUM,Unexpected generate message response reference number,,NNN)

JRST TST11G

```

;TEST DATA FIELD FOR CORRECT DATA.

```

TST11D: SETZM T1
MOVE S1,RETBUF
MOVE S2,[POINT 8,3(S1),15] ; SET POINTER IN S2.

TST11E: ILDB T4,S2 ; DATA INT T4.
CAMN T4,GNCNST ; COMPARE DATA TO EXPECTED.
JRST TST11F ; COMPARE IS GOOD.
MOVE T2,GNCNST ; T2=DATA EXPECTED.
ERROR1 (1,T2,T4,Data compare error,,NNNBYT) ; EXIT AFTER ERROR.
JRST TST11G

```

;TEST IF DATA COMPARE COMPLETED.

```

TST11F: ADDI T1,1
CAMI T1,ITERCT ; TEST BYTE COUNT = DONE.
JRST TST11E ; BYTE COUNT NOT DONE,CONTINUE TESTING.

```

;Requeue the buffer

```

TST11G: $CALL REQUEM

```


541
 542
 543
 544
 545
 546
 547 000322' 400 01 0 00 000000
 548 000323' 201 02 0 00 012330'
 549 000324' 260 17 0 00 011372'
 550
 551
 552
 553 000325' 201 01 0 00 000000
 554 000326' 202 01 0 00 000313*
 555
 556 000327' 201 01 0 00 000001
 557 000330' 202 01 0 00 000254*
 558
 559 000331' 260 17 0 00 000257*
 560
 561
 562
 563 000332' 260 17 0 00 011111'
 564 000333' 254 00 0 00 000342'
 565
 566 000334' 260 17 0 00 012114'
 567 000335' 254 00 0 00 000535'
 568 000336' 260 17 0 00 012137'
 569 000337' 254 00 0 00 000535'
 570 000340' 260 17 0 00 012163'
 571 000341' 254 00 0 00 000535'
 572
 573
 574
 575 000342' 402 00 0 00 011430'
 576 000343' 476 00 0 00 011422'
 577 000344' 260 17 0 00 011043'
 578
 579 000345' 254 00 0 00 000356'
 580 000346' 260 17 0 00 012203'
 581 000347' 254 00 0 00 000402'
 582 000350' 260 17 0 00 012223'
 583 000351' 254 00 0 00 000402'
 584 000352' 260 17 0 00 012243'
 585 000353' 254 00 0 00 000402'
 586
 587 000354' 260 17 0 00 012340'
 588 000355' 254 00 0 00 000402'

```

;*****
;* TST11A - MESSAGE TEST. THIS TEST WILL SEND A MESSAGE REQUEST OF
;* DATA=BYTE PAIR, LENGTH=4, DELAY=0, REPEAT COUNT=0.
;*****

TST11H: SETZ    S1,                ;NO TEST CHANGE
        MOVEI   S2,[ASCIZ/Message Test (Count=4, Data=Byte Pair)/]
        $CALL   TRNODE            ;Update test

;Build and send generate message ctp packet

        MOVEI   S1,0
        MOVEM   S1,GNCNST          ; GENCONST=0.

        MOVEI   S1,1
        MOVEM   S1,GENFUN          ; GENFUNCT =1/GENFBPAIR

        $CALL   BLDCTP

;TRANSMIT REQUEST MESSAGE

        $CALL   XMTREQ             ;Transmit request
        JRST    TST11I            ;OK
        ERROR1  (,,, <Timed out waiting for event pending after transmitting generate message
>,,NNN)
        JRST    TST11X
        ERROR1  (,,, <No send complete event after transmitting generate message>,,NNN)
        JRST    TST11X
        ERROR1  (,,, <Timed out waiting for generate message response available flag>,,NNN)
        JRST    TST11X

;RECEIVE RESPONSE MESSAGE

TST11I: SETZM   EXPSTA
        SETOM   CNTFLG
        $CALL   RCVRM

        JRST    TST11J
        ERROR1  (-1,EXPRSP,RCVRSP,Unexpected generate message response opcode,,NNN)
        JRST    TST11I
        ERROR1  (-1,EXPSTA,RCVSTA,Unexpected generate message response status,,NNN)
        JRST    TST11I
        ERROR1  (-1,EXPACT,RCVACT,Unexpected generate message response count,,NNN)
        JRST    TST11I
        ERROR1  (1,NUMREF,RNVNUM,Unexpected generate message response reference number,,NNN)

        JRST    TST11I

```

589
 590
 591
 592 000356' 402 00 0 00 000003
 593 000357' 200 01 0 00 000306*
 594 000360' 200 06 0 00 012314'
 595 000361' 403 10 0 00 000011
 596 000362' 254 00 0 00 000366'
 597
 598
 599
 600 000363' 271 11 0 00 000001
 601 000364' 622 11 0 00 000400
 602 000365' 271 10 0 00 000001
 603
 604
 605
 606 000366' 271 03 0 00 000001
 607 000367' 134 04 0 00 000006
 608 000370' 134 02 0 00 000006
 609 000371' 306 11 0 04 000000
 610 000372' 254 00 0 00 000374'
 611 000373' 260 17 0 00 012360'
 612
 613 000374' 271 03 0 00 000001
 614 000375' 306 10 0 02 000000
 615 000376' 254 00 0 00 000400'
 616 000377' 260 17 0 00 012400'
 617
 618
 619
 620 000400' 312 03 0 00 011510'
 621 000401' 254 00 0 00 000363'
 622
 623
 624
 625 000402' 260 17 0 00 000321*

;TEST DATA PAIR = 000,000/001,000

TST11J: SETZM T1 ; T1=BYTE COUNTER WITHIN BUFFER.
 MOVE S1,RETBUF ; S1=ADRS OF BUFFER.
 MOVE T4,[POINT 8,3(S1),15] ; T4=POINTER TO DATA.
 CLEARB P2,P3 ; CLEAR COMPARAND REGS.
 JRST TST11L

;INCREMENT COMPARAND REGISTERS.

TST11K: ADDI P3,1 ; INCREMENT L/S BYTE OF BYTEPAIR.
 TRZE P3,400 ; TEST IF INCREMENT M/S BYTE NECESSARY.
 ADDI P2,1 ; INCREMENT M/S BYTE.

;COMPARE DATA BYTES.

TST11L: ADDI T1,1 ; INCR PRESENT BYTE COUNTER.
 ILDB T2,T4 ; T2=L/S BYTE DATA.
 ILDB S2,T4 ; S2=M/S BYTE DATA.
 CAIN P3,0(T2) ; COMPARE L/S BYTE.
 JRST TST11M
 ERROR1 (1,P3,T2,Data compare error on least significant byte,,NNNBYT)

TST11M: ADDI T1,1 ; INCR PRESENT BYTE COUNTER.
 CAIN P2,0(S2) ; COMPARE
 JRST TST11N ; EXIT
 ERROR1 (1,P2,S2,Data compare error on most significant byte,,NNNBYT)

;TEST IF DATA COMPARE COMPLETED.

TST11N: CAME T1,ITERCT ; TEST BYTE COUNT = DONE.
 JRST TST11K ; BYTE COUNT NOT DONE,CONTINUE TESTING.

;Requeue the buffer

TST11O: \$CALL REQUEM

```

626
627
628
629
630
631
632 000403' 201 01 0 00 000000
633 000404' 201 02 0 00 012457'
634 000405' 260 17 0 00 011372'
635
636
637
638 000406' 201 01 0 00 000002
639 000407' 202 01 0 00 000330*
640
641 000410' 260 17 0 00 000331*
642
643
644
645 000411' 260 17 0 00 011111'
646 000412' 254 00 0 00 000421'
647
648 000413' 260 17 0 00 012437'
649 000414' 254 00 0 00 000535'
650 000415' 260 17 0 00 012137'
651 000416' 254 00 0 00 000535'
652 000417' 260 17 0 00 012163'
653 000420' 254 00 0 00 000535'
654
655
656
657 000421' 402 00 0 00 011430'
658 000422' 476 00 0 00 011422'
659 000423' 260 17 0 00 011043'
660
661 000424' 254 00 0 00 000435'
662 000425' 260 17 0 00 012203'
663 000426' 254 00 0 00 000447'
664 000427' 260 17 0 00 012223'
665 000430' 254 00 0 00 000447'
666 000431' 260 17 0 00 012243'
667 000432' 254 00 0 00 000447'
668
669 000433' 260 17 0 00 012446'
670 000434' 254 00 0 00 000447'

```

```

;*****
;* TST11B - MESSAGE TEST. THIS TEST WILL SEND A MESSAGE REQUEST OF
;* DATA=RESPONDER NODE, LENGTH=4, DELAY=0, REPEAT COUNT=0.
;*****

        MOVEI    S1,0                ;No change in test
        MOVEI    S2,ASCIZ/Message Test (Count=4, Data=Responder Node)/]
        $CALL    TRNODE              ;Change message

;Build generate message ctp packet

        MOVEI    S1,2
        MOVEM    S1,GENFUN           ; GENFUNCT =2/GENFNODE

        $CALL    BLDCTP

;TRANSMIT REQUEST MESSAGE

        $CALL    XMTREQ              ;Transmit request
        JRST     TST11P              ;OK
        ERROR1    (,,, <Timed out waiting for event pending after transmitting generate message
>,,NNN)
        JRST     TST11X
        ERROR1    (,,, <No send complete event after transmitting generate message>,,NNN)
        JRST     TST11X
        ERROR1    (,,, <Timed out waiting for generate message response available flag>,,NNN)
        JRST     TST11X

;RECEIVE RESPONSE MESSAGE

TST11P: SETZM    EXPSTA
        SETOM    CNTFLG
        $CALL    RCVRM

        JRST     TST11Q
        ERROR1    (-1,EXPRSP,RCVRSP,Unexpected generate message response opcode,,NNN)
        JRST     TB11Z
        ERROR1    (-1,EXPSTA,RCVSTA,Unexpected generate message response status,,NNN)
        JRST     TB11Z
        ERROR1    (-1,EXPACT,RCVACT,Unexpected generate message response count,,NNN)
        JRST     TB11Z
        ERROR1    (1,NUMREF,RNVNUM,Unexpected generate message response reference number,,NNN)

        JRST     TB11Z

```

671
 672
 673
 674
 675 000435' 402 00 0 00 000003
 676 000436' 200 01 0 00 000357*
 677 000437' 200 02 0 00 012314'
 678
 679 000440' 271 03 0 00 000001
 680 000441' 134 06 0 00 000002
 681 000442' 312 06 0 00 000000*
 682 000443' 260 17 0 00 012455'
 683
 684
 685
 686 000444' 312 03 0 00 011510'
 687 000445' 254 00 0 00 000440'
 688 000446' 254 00 0 00 000447'
 689
 690
 691
 692 000447' 260 17 0 00 000402*
 693

```
;TEST DATA FIELD
;S1=DATA HOLDING REG,S2=DATA POINTER REG,T1=BYTE COUNTER

TST11Q: SETZM  T1           ; T1=BYTE COUNTER OF DATA READ BACK.
        MOVE   S1,RETBUF
        MOVE   S2,[POINT 8,3(S1),15] ; 1ST DATA BYTE TO BE EXAMINED.

TST11R: ADDI    T1,1         ; INCR BYTE COUNT.
        ILDB   T4,S2        ; LOAD DATA BYTE FROM BUFFER
        CAME   T4,NODEN     ; COMPARE TO NODE NUMBER.
        ERROR1 (1,NODEN,T4,Data compare error,,NNBYT)

;TEST IF ALL BYTES DONE.

        CAME   T1,ITERCT    ; TEST IF ALL BUFFER TESTED.
        JRST  TST11R       ; CONTINUE IN DATA COMPARE LOOP.
        JRST  TB11Z        ; LOOP DONE.

;Requeue the buffer
TB11Z:  $CALL  REQUEM
```



```

694
695
696
697
698
699
700 000450' 201 01 0 00 000000
701 000451' 201 02 0 00 012464'
702 000452' 260 17 0 00 011372'
703
704
705
706 000453' 201 01 0 00 000004
707 000454' 202 01 0 00 011510'
708
709 000455' 201 01 0 00 000003
710 000456' 202 01 0 00 000407*
711
712 000457' 200 01 0 00 012474'
713 000460' 202 01 0 00 011433'
714 000461' 201 01 0 00 000004
715 000462' 202 01 0 00 011432'
716
717 000463' 260 17 0 00 000410*
718
719
720
721 000464' 260 17 0 00 011111'
722 000465' 254 00 0 00 000474'
723
724 000466' 260 17 0 00 012437'
725 000467' 254 00 0 00 000535'
726 000470' 260 17 0 00 012137'
727 000471' 254 00 0 00 000535'
728 000472' 260 17 0 00 012163'
729 000473' 254 00 0 00 000535'
730
731
732
733 000474' 402 00 0 00 011430'
734 000475' 476 00 0 00 011422'
735 000476' 260 17 0 00 011043'
736
737 000477' 254 00 0 00 000510'
738 000500' 260 17 0 00 012203'
739 000501' 254 00 0 00 000534'
740 000502' 260 17 0 00 012223'
741 000503' 254 00 0 00 000534'
742 000504' 260 17 0 00 012243'
743 000505' 254 00 0 00 000534'
744
745 000506' 260 17 0 00 012475'
746 000507' 254 00 0 00 000534'
  
```

```

*****
* TST11C - MESSAGE TEST. THIS TEST WILL SEND A MESSAGE REQUEST OF
* DATA=IMAGE DATA, LENGTH=4, DELAY=0, REPEAT COUNT=0.
*****
  
```

```

MOVEI S1,0 ;No change in test
MOVEI S2,[ASCIZ/Message Test (Count=4, Data=Image Data)/]
SCALL TRNODE ;Change message
  
```

;Build generate message ctp packet

```

MOVEI S1,4
MOVEM S1,ITERCT ; NEED FOR GENMSG RTN.

MOVEI S1,3
MOVEM S1,GENFUN ; GENFUNCT =3/GENFIMAGE

MOVE S1,[777773757700] ; IMAGE=377,376,375,374.
MOVEM S1,IMAGDT
MOVEI S1,4
MOVEM S1,IMGLN ; IMAGE DATA LENGTH
  
```

SCALL BLDCTP

;TRANSMIT REQUEST MESSAGE

```

SCALL XMTREQ ;Transmit request
JRST TST11S ;OK
ERROR1 (,,, <Timed out waiting for event pending after transmitting generate message
>,,NNN)
JRST TST11X
ERROR1 (,,, <No send complete event after transmitting generate message>,,NNN)
JRST TST11X
ERROR1 (,,, <Timed out waiting for generate message response available flag>,,NNN)
JRST TST11X
  
```

;RECEIVE RESPONSE MESSAGE

```

TST11S: SETZM EXPSTA
SETOM CNTFLG
SCALL RCVRM

JRST TST11T
ERROR1 (-1,EXPRSP,RCVRSP,Unexpected generate message response opcode,,NNN)
JRST TST11W
ERROR1 (-1,EXPSTA,RCVSTA,Unexpected generate message response status,,NNN)
JRST TST11W
ERROR1 (-1,EXPACT,RCVACT,Unexpected generate message response count,,NNN)
JRST TST11W
ERROR1 (1,NUMREF,RNVNUM,Unexpected generate message response reference number,,NNN)

JRST TST11W
  
```


747
 748
 749
 750
 751
 752
 753
 754
 755
 756 000510' 200 01 0 00 000436*
 757 000511' 200 02 0 00 012314'
 758 000512' 201 03 0 00 000000
 759 000513' 201 05 0 00 011433'
 760 000514' 275 05 0 00 000001
 761 000515' 200 06 0 00 012504'
 762 000516' 201 07 0 00 000000
 763
 764
 765
 766 000517' 134 04 0 00 000006
 767 000520' 271 07 0 00 000001
 768 000521' 312 07 0 00 011432'
 769 000522' 254 00 0 00 000525'
 770 000523' 201 07 0 00 000000
 771 000524' 200 06 0 00 012504'
 772
 773
 774
 775 000525' 134 10 0 00 000002
 776
 777
 778
 779 000526' 302 10 0 04 000000
 780 000527' 260 17 0 00 012505'
 781
 782
 783
 784 000530' 271 03 0 00 000001
 785 000531' 311 03 0 00 011510'
 786 000532' 254 00 0 00 000534'
 787 000533' 254 00 0 00 000517'
 788
 789
 790
 791 000534' 260 17 0 00 000447*

;DATA COMPARISON ROUTINE FOR IMAGDT.
 ; REGISTER USAGE FOR BUFFER DATA:
 ; P2=DATA HOLDING REG OF BUFFER
 ; S2=BYTE POINTER FOR BUFFER
 ; REGISTER USAGE FOR IMAGE DATA:
 ; T2=DATA HOLDING REG OF IMAGE DATA
 ; T4=BYTE POINTER FOR IMAGE DATA

S1=ADDRESS OF BUFFER
 T1=BYTE COUNTER OF BUFFER

T3=ADDRESS OF IMAGDT
 P1=BYTE COUNTER OF IMAGDT

TST11T: MOVE S1,RETBUF
 MOVE S2,[POINT 8,3(S1),15] ; 1ST DATA BYTE TO BE EXAMINED-1.
 MOVEI T1,0 ; RESET BUFFER BYTE COUNTER
 MOVEI T3,IMAGDT
 SUBI T3,1
 MOVE T4,[POINT 8,(T3),31] ; 1ST IMAGE BYTE TO BE EXAMINED-1.
 MOVEI P1,0 ; RESET IMAGE BYTE COUNTER.

;GET IMAGE DATA BYTE.

TST11U: ILDB T2,T4 ; T2 = IMAGE DATA BYTE.
 ADDI P1,1 ; INCR IMAGE DATA COUNT.
 CAML P1,IMGLN ; TEST IF IMAGE LENGTH DONE.
 JRST TST11V ; NO, CONTINUE WITH CURRENT COMPARE.
 MOVEI P1,0 ; RESET CURRENT IMAGE COUNT.
 MOVE T4,[POINT 8,(T3),31] ; RESET IMAGE DATA POINTER.

;GET BUFFER DATA BYTE.

TST11V: ILDB P2,S2 ; P2=BUFFER DATA BYTE

;COMPARE BUFFER DATA WITH IMAGE DATA.

CAIE P2,0(T2)
 ERROR1 (1,T2,P2,Data compare error,,NNNBYT)

;COMPARE BUFFER BYTE COUNT WITH COMMAND BYTE COUNT.

ADDI T1,1 ; INCR PRESENT BYTE POINTER
 CAML T1,ITERCT ; TEST IF PRESENT = ISSUED BYTES.
 JRST TST11W ; JUMP IF ALL BUFFER TESTED.
 JRST TST11U ; CONTINUE 'TIL ALL DATA BLOCK TESTED.

;Requeue the buffer

TST11W: \$CALL REQUEM

DFCIBM CTP Tests 01-80 version 2(20)
DFCIBM MAC 22-Aug-85 20:30

MACRO X53B(1242) 17:38 27-Aug-85 Page 18
TST11 Message Test - 4 bytes each of 4 data types

SEQ 0356

```
792  
793  
794  
795 000535' 260 17 0 00 000222*  
796  
797  
798  
799 000536' 260 17 0 00 011306'  
800 000537' 254 00 0 00 000542'  
801 000540' 260 17 0 00 011650'  
802 000541' 254 00 0 00 000544'  
803  
804  
805  
806 000542' 201 01 0 00 000764  
807 000543' 260 17 0 00 000230*  
808 000544' 263 17 0 00 000000
```

;Disconnect with selected node

TST11X: \$CALL DODIS ;Disconnect with the selected node

;Wait for connection closed

\$CALL CONCLD
JRST TST11Y
ERROR1 (Connection closed not in disconnect status,,NNN)
JRST TST11Z

;Wait

TST11Y: MOVEI S1,^D500
\$CALL WAITX
TST11Z: \$RET

SUBTTL TST12 Message Test - 4 bytes each of 4 data types (repeat=3)

 ;* TST12 - MESSAGE TEST. THIS TEST WILL SEND A MESSAGE REQUEST OF
 ; DATA=376, LENGTH=4, DELAY=0, REPEAT COUNT=2.
 ;*****

809
 810
 811
 812
 813
 814
 815
 816 000545' 201 01 0 00 012514'
 817 000546' 201 02 0 00 012516'
 818 000547' 260 17 0 00 011372'
 819
 820
 821
 822 000550' 260 17 0 00 011134'
 823 000551' 254 00 0 00 000560'
 824
 825 000552' 260 17 0 00 011562'
 826 000553' 254 00 0 00 001525'
 827
 828 000554' 260 17 0 00 011607'
 829 000555' 254 00 0 00 001525'
 830 000556' 260 17 0 00 011630'
 831 000557' 254 00 0 00 001525'
 832
 833
 834
 835 000560' 201 01 0 00 000004
 836 000561' 202 01 0 00 000246*
 837 000562' 202 01 0 00 000247*
 838 000563' 202 01 0 00 011510'
 839
 840 000564' 201 01 0 00 000000
 841 000565' 202 01 0 00 000456*
 842 000566' 202 01 0 00 000252*
 843
 844 000567' 201 01 0 00 000376
 845 000570' 202 01 0 00 000326*
 846
 847 000571' 201 01 0 00 000002
 848 000572' 202 01 0 00 000253*
 849
 850 000573' 260 17 0 00 000463*
 851
 852
 853
 854 000574' 260 17 0 00 011111'
 855 000575' 254 00 0 00 000604'
 856
 857 000576' 260 17 0 00 012437'
 858 000577' 254 00 0 00 001520'
 859 000600' 260 17 0 00 012137'
 860 000601' 254 00 0 00 001520'
 861 000602' 260 17 0 00 012163'
 862 000603' 254 00 0 00 001520'

TST12: MOVEI S1,[ASCIZ/TST12/] ;Load test name
 MOVEI S2,[ASCIZ/Message Test (Count=4, Data=4 Bytes Of 376 Repeated 3 Times)/]
 \$CALL TRNODE ;Print node number if trace switch set

 ;Connect with selected node and verify connection=open.

 \$CALL CONNCT ;CONNECT with the selected node
 JRST T12B
 ERROR1 (,,,Timed out waiting for event pending flag during connection attempt,,NNN)

 JRST TST12Y
 ERROR1 (,,,Event pending was not connection accepted during connection attempt,,NNN)
)
 JRST TST12Y
 ERROR1 (,,,Connection not open after connection accepted,,NNN)
 JRST TST12Y

 ;Build generate message ctp packet
 T12B: MOVEI S1,4
 MOVEM S1,OPCODE ; OPCODE = GEN MESSAGE.
 MOVEM S1,GENLEN ; GEN LENGTH=4.
 MOVEM S1,ITERCT ; ITERCT=4.

 MOVEI S1,0
 MOVEM S1,GENFUNCT ; GENFUNCT =0/GENFILL.
 MOVEM S1,DELAY ; DELAY = 0

 MOVEI S1,376
 MOVEM S1,GENCONST ; GENCONST=376.

 MOVEI S1,2
 MOVEM S1,RCOUNT ; REPEAT COUNT = 2.

 \$CALL BLDCTP

 ;TRANSMIT REQUEST MESSAGE

 \$CALL XMTREQ ;Transmit request
 JRST T12C ;OK
 ERROR1 (,,,<Timed out waiting for event pending after transmitting generate message
 >,,NNN)
 JRST TST12X
 ERROR1 (,,,<No send complete event after transmitting generate message>,,NNN)
 JRST TST12X
 ERROR1 (,,,<Timed out waiting for generate message response available flag>,,NNN)
 JRST TST12X

863
 864
 865
 866 000604' 402 00 0 00 011430'
 867 000605' 402 00 0 00 011423'
 868 000606' 476 00 0 00 011422'
 869 000607' 260 17 0 00 011043'
 870 000610' 254 00 0 00 000621'
 871
 872 000611' 260 17 0 00 012536'
 873 000612' 254 00 0 00 000635'
 874 000613' 260 17 0 00 012554'
 875 000614' 254 00 0 00 000635'
 876
 877 000615' 260 17 0 00 012563'
 878 000616' 254 00 0 00 000635'
 879
 880 000617' 260 17 0 00 012572'
 881 000620' 254 00 0 00 000635'
 882
 883
 884
 885 000621' 402 00 0 00 000003
 886 000622' 200 01 0 00 000510*
 887 000623' 200 02 0 00 012314'
 888 000624' 134 06 0 00 000002
 889 000625' 316 06 0 00 000570*
 890 000626' 254 00 0 00 000632'
 891 000627' 200 04 0 00 000625*
 892 000630' 260 17 0 00 012321'
 893 000631' 254 00 0 00 000635'
 894
 895
 896
 897 000632' 271 03 0 00 000001
 898 000633' 312 03 0 00 011510'
 899 000634' 254 00 0 00 000624'
 900
 901
 902
 903 000635' 260 17 0 00 000534*
 904
 905
 906
 907 000636' 260 17 0 00 011330'
 908 000637' 254 00 0 00 000642'
 909 000640' 260 17 0 00 012613'
 910 000641' 254 00 0 00 001520'

;RECEIVE RESPONSE MESSAGE

T12C: SETZM EXPSTA
 SETZM EXPACT
 SETOM CNTFLG
 \$CALL RCVRM
 JRST T12D
 ERROR1 (-1,EXPRSP,RCVRSP,Unexpected generate message response opcode,actual count=0
 ,NNN)
 JRST T12G
 ERROR1 (-1,EXPSTA,RCVSTA,Unexpected generate message status,actual count=0,NNN)
 JRST T12G
 ERROR1 (-1,EXPACT,RCVACT,Unexpected generate message response count,actual count=0,
 NNN)
 JRST T12G
 ERROR1 (1,NUMREF,RNVNUM,Unexpected generate message response reference number,actua
 l count=0,NNN)
 JRST T12G

;TEST DATA FIELD FOR CORRECT DATA.

T12D: SETZM T1
 MOVE S1,RETBUF
 MOVE S2,[POINT 8,3(S1),15] ; SET POINTER IN S2.
 T12E: ILDB T4,S2 ; DATA INT T4.
 CAMN T4,GNCNST ; COMPARE DATA TO EXPECTED.
 JRST T12F ; COMPARE IS GOOD.
 MOVE T2,GNCNST ; T2=DATA EXPECTED.
 ERROR1 (1,T2,T4,Data compare error,,NNNBYT)
 JRST T12G ; EXIT FROM TEST.

;TEST IF DATA COMPARE COMPLETED.

T12F: ADDI T1,1
 CAME T1,ITERCT ; TEST BYTE COUNT = DONE.
 JRST T12E ; BYTE COUNT NOT DONE,CONTINUE TESTING.

;Requeue the buffer after 1st message received and tested.

T12G: \$CALL REQUEM

;Wait for message available

\$CALL MSGAVL ; WAIT FOR MESSAGE AVAILABLE FLAG
 JRST T12H ; NO FAILURE,CONTINUE
 ERROR1 (,,,Time out waiting for generate message response,,NNN)
 JRST TST12X

911
 912
 913
 914 000642' 201 01 0 00 000400
 915 000643' 202 01 0 00 011423'
 916 000644' 260 17 0 00 011043'
 917 000645' 254 00 0 00 000656'
 918
 919 000646' 260 17 0 00 012625'
 920 000647' 254 00 0 00 000672'
 921
 922 000650' 260 17 0 00 012634'
 923 000651' 254 00 0 00 000672'
 924
 925 000652' 260 17 0 00 012643'
 926 000653' 254 00 0 00 000672'
 927
 928 000654' 260 17 0 00 012652'
 929 000655' 254 00 0 00 000672'
 930
 931
 932
 933 000656' 402 00 0 00 000003
 934 000657' 200 01 0 00 000622*
 935 000660' 200 02 0 00 012314'
 936 000661' 134 06 0 00 000002
 937 000662' 316 06 0 00 000627*
 938 000663' 254 00 0 00 000667*
 939 000664' 200 04 0 00 000662*
 940 000665' 260 17 0 00 012321'
 941 000666' 254 00 0 00 000672'
 942
 943
 944
 945 000667' 271 03 0 00 000001
 946 000670' 312 03 0 00 011510'
 947 000671' 254 00 0 00 000661'
 948
 949
 950
 951 000672' 260 17 0 00 000635*
 952
 953
 954
 955 000673' 260 17 0 00 011330'
 956 000674' 254 00 0 00 000677'
 957 000675' 260 17 0 00 012613'
 958 000676' 254 00 0 00 001520'

;Recieve response, test act count = 1.

T12H: MOVEI S1,1*^D256
 MOVEM S1,EXPACT
 \$CALL RCVRM
 JRST T12I
 ERROR1 (-1,EXPRSP,RCVRSP,Unexpected generate message response opcode,actual count=1
 ,NNN)
 JRST T12L
 ERROR1 (-1,EXPSTA,RCVSTA,Unexpected generate message response status,actual count=1
 ,NNN)
 JRST T12L
 ERROR1 (-1,EXPACT,RCVACT,Unexpected generate message response count,actual count=1,
 NNN)
 JRST T12L
 ERROR1 (1,NUMREF,RNVNUM,Unexpected generate message response reference number,actua
 l count=1,NNN)
 JRST T12L

;TEST DATA FIELD FOR CORRECT DATA.

T12I: SETZM T1
 MOVE S1,RETBUF
 MOVE S2,[POINT 8,3(S1),15] ; SET POINTER IN S2.
 T12J: ILDB T4,S2 ; DATA INT T4.
 CAMN T4,GNCNST ; COMPARE DATA TO EXPECTED.
 JRST T12K ; COMPARE IS GOOD.
 MOVE T2,GNCNST ; T2=DATA EXPECTED.
 ERROR1 (1,T2,T4,Data compare error,,NNNBYT)
 JRST T12L ; EXIT FROM TEST.

;TEST IF DATA COMPARE COMPLETED.

T12K: ADDI T1,1
 CAME T1,ITERCT ; TEST BYTE COUNT = DONE.
 JRST T12J ; BYTE COUNT NOT DONE,CONTINUE TESTING.

;Requeue the buffer after 2nd message received and tested.

T12L: \$CALL REQUEM

;WAIT FOR 3RD MESSAGE AVAILABLE

\$CALL MSGAVL ; WAIT FOR MESSAGE AVAILABLE FLAG
 JRST T12M ; NO FAILURE,CONTINUE
 ERROR1 (,Time out waiting for generate message response,,NNN)
 JRST T12X

959
 960
 961
 962 000677' 201 01 0 00 001000
 963 000700' 202 01 0 00 011423'
 964 000701' 260 17 0 00 011043'
 965
 966 000702' 254 00 0 00 000713'
 967
 968 000703' 260 17 0 00 012664'
 969 000704' 254 00 0 00 000727'
 970
 971 000705' 260 17 0 00 012673'
 972 000706' 254 00 0 00 000727'
 973
 974 000707' 260 17 0 00 012702'
 975 000710' 254 00 0 00 000727'
 976
 977 000711' 260 17 0 00 012711'
 978 000712' 254 00 0 00 000727'
 979
 980
 981
 982 000713' 402 00 0 00 000003
 983 000714' 200 01 0 00 000657*
 984 000715' 200 02 0 00 012314'
 985
 986 000716' 134 06 0 00 000002
 987 000717' 316 06 0 00 000664*
 988 000720' 254 00 0 00 000724*
 989 000721' 200 04 0 00 000717*
 990 000722' 260 17 0 00 012321*
 991 000723' 254 00 0 00 000727*
 992
 993
 994
 995 000724' 271 03 0 00 000001
 996 000725' 312 03 0 00 011510*
 997 000726' 254 00 0 00 000716*
 998
 999
 1000
 1001 000727' 260 17 0 00 000672*
 1002

;Recieve response, test act count = 2.

T12M: MOVEI S1,2*^D256
 MOVEM S1,EXPACT
 \$CALL RCVRM
 JRST T12N
 ERROR1 (-1,EXPRSP,RCVRSP,Unexpected generate message response opcode,actual count=2
 ,NNN)
 JRST T12Q
 ERROR1 (-1,EXPSTA,RCVSTA,Unexpected generate message response status,actual count=2
 ,NNN)
 JRST T12Q
 ERROR1 (-1,EXPACT,RCVACT,Unexpected generate message response count,actual count=2,
 NNN)
 JRST T12Q
 ERROR1 (1,NUMREF,RNVNUM,Unexpected generate message response reference number,actua
 l count=2,NNN)
 JRST T12Q

;TEST DATA FIELD FOR CORRECT DATA.

T12N: SETZM T1
 MOVE S1,RETBUR
 MOVE S2,[POINT 8,3(S1),15] ; SET POINTER IN S2.
 T12O: ILDB T4,S2 ; DATA INT T4.
 CAMN T4,GNCNST ; COMPARE DATA TO EXPECTED.
 JRST T12P ; COMPARE IS GOOD.
 MOVE T2,GNCNST ; T2=DATA EXPECTED.
 ERROR1 (1,T2,T4,Data compare error,,NNNBYT)
 JRST T12Q ; EXIT FROM TEST.

;TEST IF DATA COMPARE COMPLETED.

T12P: ADDI T1,1
 CAME T1,ITERCT ; TEST BYTE COUNT = DONE.
 JRST T12Q ; BYTE COUNT NOT DONE,CONTINUE TESTING.

;Requeue the buffer after the 3rd message received and tested.

T12Q: \$CALL REQUEM

```

1003
1004
1005
1006
1007
1008
1009 000730' 201 01 0 00 000000
1010
1011 000731' 201 02 0 00 012720'
1012 000732' 260 17 0 00 011372'
1013
1014
1015
1016 000733' 201 01 0 00 000004
1017 000734' 202 01 0 00 000561*
1018 000735' 202 01 0 00 000562*
1019 000736' 202 01 0 00 011510'
1020
1021 000737' 201 01 0 00 000000
1022 000740' 202 01 0 00 000566*
1023 000741' 202 01 0 00 000721*
1024
1025 000742' 201 01 0 00 000002
1026 000743' 202 01 0 00 000572*
1027
1028 000744' 201 01 0 00 000001
1029 000745' 202 01 0 00 000565*
1030
1031 000746' 260 17 0 00 000573*
1032
1033
1034
1035 000747' 260 17 0 00 011111'
1036 000750' 254 00 0 00 000757'
1037
1038 000751' 260 17 0 00 012437'
1039 000752' 254 00 0 00 001520'
1040 000753' 260 17 0 00 012137'
1041 000754' 254 00 0 00 001520'
1042 000755' 260 17 0 00 012163'
1043 000756' 254 00 0 00 001520'

```

```

*****
* TST12A - MESSAGE TEST. THIS TEST WILL SEND A MESSAGE REQUEST OF
* DATA=BYTE PAIR, LENGTH=4, DELAY=0, REPEAT COUNT=2.
*****

TST12A: MOVEI S1,0 ;Load zero for no message
        MOVEI S2,[ASCIZ/Message Test (Count=4, Data=4 Bytes Of Byte Pair Repeated 3 Times/
        ]
        $CALL TRNODE

;Build generate message ctp packet

        MOVEI S1,4
        MOVEM S1,OPCODE ; OPCODE = GEN MESSAGE.
        MOVEM S1,GENLEN ; GEN LENGTH=4.
        MOVEM S1,ITERCT ; ITERCT=4.

        MOVEI S1,0
        MOVEM S1,DELAY ; DELAY = 0
        MOVEM S1,GENCONST ; GENCONST=0.

        MOVEI S1,2
        MOVEM S1,RCOUNT ; REPEAT COUNT = 2.

        MOVEI S1,1
        MOVEM S1,GENFUNCT ; GENFUNCT =1/GENFBPAIR

        $CALL BLDCTP

;TRANSMIT REQUEST MESSAGE

        $CALL XMTREQ ;Transmit request
        JRST T12AA ;OK
        ERROR1 (,,, <Timed out waiting for event pending after transmitting generate message
>,,NNN)
        JRST TST12X
        ERROR1 (,,, <No send complete event after transmitting generate message>,,NNN)
        JRST TST12X
        ERROR1 (,,, <Timed out waiting for generate message response available flag>,,NNN)
        JRST TST12X

```

1044
 1045
 1046
 1047 000757' 402 00 0 00 011423'
 1048 000760' 402 00 0 00 011430'
 1049 000761' 476 00 0 00 011422'
 1050 000762' 260 17 0 00 011043'
 1051
 1052 000763' 254 00 0 00 000774'
 1053
 1054 000764' 260 17 0 00 012536'
 1055 000765' 254 00 0 00 001020'
 1056
 1057 000766' 260 17 0 00 012736'
 1058 000767' 254 00 0 00 001020'
 1059
 1060 000770' 260 17 0 00 012563'
 1061 000771' 254 00 0 00 001020'
 1062
 1063 000772' 260 17 0 00 012745'
 1064 000773' 254 00 0 00 001020'
 1065
 1066
 1067
 1068 000774' 402 00 0 00 000003
 1069 000775' 200 01 0 00 000714*
 1070 000776' 200 06 0 00 012314'
 1071 000777' 403 10 0 00 000011
 1072 001000' 254 00 0 00 001004'
 1073
 1074
 1075
 1076 001001' 271 11 0 00 000001
 1077 001002' 622 11 0 00 000400
 1078 001003' 271 10 0 00 000001
 1079
 1080
 1081
 1082 001004' 271 03 0 00 000001
 1083 001005' 134 04 0 00 000006
 1084 001006' 134 02 0 00 000006
 1085 001007' 306 11 0 04 000000
 1086 001010' 254 00 0 00 001012'
 1087 001011' 260 17 0 00 012360'
 1088
 1089 001012' 271 03 0 00 000001
 1090 001013' 306 10 0 02 000000
 1091 001014' 254 00 0 00 001016'
 1092 001015' 260 17 0 00 012400'
 1093
 1094
 1095
 1096 001016' 312 03 0 00 011510'
 1097 001017' 254 00 0 00 001001'

;RECEIVE 1ST RESPONSE MESSAGE

T12AA: SETZM EXPACT
 SETZM EXPSTA
 SETOM CNTFLG
 SCALL RCVRM
 JRST T12AB
 ERROR1 (-1,EXPRSP,RCVRSP,Unexpected generate message response opcode,actual count=0
 ,NNN)
 JRST T12AG
 ERROR1 (-1,EXPSTA,RCVSTA,Unexpected generate message response status,actual count=0
 ,NNN)
 JRST T12AG
 ERROR1 (-1,EXPACT,RCVACT,Unexpected generate message response count,actual count=0,
 NNN)
 JRST T12AG
 ERROR1 (1,NUMREF,RNVNUM,Unexpected generate message response reference number,actua
 l count=0,NNN)
 JRST T12AG

;TEST DATA PAIR = 000,000/001,000

T12AB: SETZM T1 ; T1=BYTE COUNTER WITHIN BUFFER.
 MOVE S1,RETBUF ; S1=ADRS OF BUFFER.
 MOVE T4,[POINT 8,3(S1),15] ; T4=POINTER TO DATA.
 CLEARB P2,P3 ; CLEAR COMPARAND REGS.
 JRST T12AD

;INCREMENT COMPARAND REGISTERS.

T12AC: ADDI P3,1 ; INCREMENT L/S BYTE OF BYTEPAIR.
 TRZE P3,400 ; TEST IF INCREMENT M/S BYTE NECESSARY.
 ADDI P2,1 ; INCREMENT M/S BYTE.

;COMPARE DATA BYTES.

T12AD: ADDI T1,1 ; INCR PRESENT BYTE COUNTER.
 ILDB T2,T4 ; T2=L/S BYTE DATA.
 ILDB S2,T4 ; S2=M/S BYTE DATA.
 CAIN P3,0(T2) ; COMPARE L/S BYTE.
 JRST T12AE
 ERROR1 (1,P3,12,Data compare error on least significant byte,,NNNBYT)

T12AE: ADDI T1,1 ; INCR PRESENT BYTE COUNTER.
 CAIN P2,0(S2) ; COMPARE
 JRST T12AF ; EXIT
 ERROR1 (1,P2,S2,Data compare error on most significant byte,,NNNBYT)

;TEST IF DATA COMPARE COMPLETED.

T12AF: CAME T1,ITERCT ; TEST BYTE COUNT = DONE.
 JRST T12AC ; BYTE COUNT NOT DONE,CONTINUE TESTING.

1098
 1099
 1100
 1101 001020' 260 17 0 00 000727*
 1102
 1103
 1104
 1105 001021' 260 17 0 00 011330'
 1106 001022' 254 00 0 00 001025'
 1107 001023' 260 17 0 00 012613'
 1108 001024' 254 00 0 00 001520'
 1109
 1110
 1111
 1112
 1113 001025' 402 00 0 00 011430'
 1114 001026' 476 00 0 00 011422'
 1115 001027' 201 01 0 00 000400'
 1116 001030' 202 01 0 00 011423'
 1117 001031' 260 17 0 00 011043'
 1118
 1119 001032' 254 00 0 00 001043'
 1120
 1121 001033' 260 17 0 00 012625'
 1122 001034' 254 00 0 00 001067'
 1123
 1124 001035' 260 17 0 00 012634'
 1125 001036' 254 00 0 00 001067'
 1126
 1127 001037' 260 17 0 00 012643'
 1128 001040' 254 00 0 00 001067'
 1129
 1130 001041' 260 17 0 00 012754'
 1131 001042' 254 00 0 00 001067'
 1132
 1133
 1134
 1135 001043' 402 00 0 00 000003'
 1136 001044' 200 01 0 00 000775*
 1137 001045' 200 06 0 00 012314'
 1138 001046' 403 10 0 00 000011'
 1139 001047' 254 00 0 00 001053'
 1140
 1141
 1142
 1143 001050' 271 11 0 00 000001'
 1144 001051' 622 11 0 00 000400'
 1145 001052' 271 10 0 00 000001'

;Requeue the buffer after the 1st message received and tested.

T12AG: \$CALL REQUEM

;WAIT FOR 2ND MESSAGE AVAILABLE

\$CALL MSGAVL ; WAIT FOR MESSAGE AVAILABLE FLAG
 JRST T12AH ; NO FAILURE,CONTINUE
 ERROR1 (,Time out waiting for generate message response,,NNN)
 JRST TST12X

;RECEIVE 2ND RESPONSE MESSAGE

;TEST RESPONSE MESSAGE OPCODE, STATUS,ACTCNT, AND REFERENCE NUMBER

T12AH: SETZM EXPSTA
 SETOM CNTFLG
 MOVEI S1,1*^D256
 MOVEM S1,EXPACT
 \$CALL RCVRM
 JRST T12AI
 ERROR1 (-1,EXPRSP,RCVRSP,Unexpected generate message response opcode,actual count=1
 ,NNN)
 JRST T12AN
 ERROR1 (-1,EXPSTA,RCVSTA,Unexpected generate message response status,actual count=1
 ,NNN)
 JRST T12AN
 ERROR1 (-1,EXPACT,RCVACT,Unexpected generate message response count,actual count=1,
 NNN)
 JRST T12AN
 ERROR1 (1,NUMREF,RNVNUM,Unexpected generate message response reference number,actua
 l count=1,NNN)
 JRST T12AN

;TEST DATA PAIR = 000,000/001,000

T12AI: SETZM T1 ; T1=BYTE COUNTER WITHIN BUFFER.
 MOVE S1,RETBUF ; S1=ADRS OF BUFFER.
 MOVE T4,[POINT 8,3(S1),15] ; T4=POINTER TO DATA.
 CLEARB P2,P3 ; CLEAR COMPARAND REGS.
 JRST T12AK

;INCREMENT COMPARAND REGISTERS.

T12AJ: ADDI P3,1 ; INCREMENT L/S BYTE OF BYTEPAIR.
 TRZE P3,400 ; TEST IF INCREMENT M/S BYTE NECESSARY.
 ADDI P2,1 ; INCREMENT M/S BYTE.

1146
 1147
 1148
 1149 001053' 271 03 0 00 000001
 1150 001054' 134 04 0 00 000006
 1151 001055' 134 02 0 00 000006
 1152 001056' 306 11 0 04 000000
 1153 001057' 254 00 0 00 001061'
 1154 001060' 260 17 0 00 012360'
 1155
 1156 001061' 271 03 0 00 000001
 1157 001062' 306 10 0 02 000000
 1158 001063' 254 00 0 00 001065'
 1159 001064' 260 17 0 00 012400'
 1160
 1161
 1162
 1163 001065' 312 03 0 00 011510'
 1164 001066' 254 00 0 00 001050'
 1165
 1166
 1167
 1168 001067' 260 17 0 00 001020*
 1169
 1170
 1171
 1172 001070' 260 17 0 00 011330'
 1173 001071' 254 00 0 00 001074'
 1174 001072' 260 17 0 00 012613'
 1175 001073' 254 00 0 00 001520'

;COMPARE DATA BYTES.

T12AK: ADDI T1,1 ; INCR PRESENT BYTE COUNTER.
 ILDB T2,T4 ; T2=L/S BYTE DATA.
 ILDB S2,T4 ; S2=M/S BYTE DATA.
 CAIN P3,0(T2) ; COMPARE L/S BYTE.
 JRST T12AL
 ERROR1 (1,P3,T2,Data compare error on least significant byte,,NNNBYT)

T12AL: ADDI T1,1 ; INCR PRESENT BYTE COUNTER.
 CAIN P2,0(S2) ; COMPARE
 JRST T12AM ; EXIT
 ERROR1 (1,P2,S2,Data compare error on most significant byte,,NNNBYT)

;TEST IF DATA COMPARE COMPLETED.

T12AM: CAME T1,ITERCT ; TEST BYTE COUNT = DONE.
 JRST T12AJ ; BYTE COUNT NOT DONE,CONTINUE TESTING.

;Requeue the buffer after the 2nd message received and tested.

T12AN: \$CALL REQUEM

;WAIT FOR 3RD MESSAGE AVAILABLE

\$CALL MSGAVL ; WAIT FOR MESSAGE AVAILABLE FLAG
 JRST T12AO ; NO FAILURE,CONTINUE
 ERROR1 (,,,Time out waiting for generate message response,,NNN)
 JRST TST12X

1176
 1177
 1178
 1179 001074' 402 00 0 00 011430'
 1180 001075' 476 00 0 00 011422'
 1181 001076' 201 01 0 00 001000
 1182 001077' 202 01 0 00 011423'
 1183 001100' 260 17 0 00 011043'
 1184 001101' 254 00 0 00 001112'
 1185
 1186 001102' 260 17 0 00 012664'
 1187 001103' 254 00 0 00 001136'
 1188
 1189 001104' 260 17 0 00 012673'
 1190 001105' 254 00 0 00 001136'
 1191
 1192 001106' 260 17 0 00 012702'
 1193 001107' 254 00 0 00 001136'
 1194
 1195 001110' 260 17 0 00 012763'
 1196 001111' 254 00 0 00 001136'
 1197
 1198
 1199
 1200 001112' 402 00 0 00 000003
 1201 001113' 200 01 0 00 001044*
 1202 001114' 200 06 0 00 012314'
 1203 001115' 403 10 0 00 000011
 1204 001116' 254 00 0 00 001122'
 1205
 1206
 1207
 1208 001117' 271 11 0 00 000001
 1209 001120' 622 11 0 00 000400
 1210 001121' 271 10 0 00 000001
 1211
 1212
 1213
 1214 001122' 271 03 0 00 000001
 1215 001123' 134 04 0 00 000006
 1216 001124' 134 02 0 00 000006
 1217 001125' 306 11 0 04 000000
 1218 001126' 254 00 0 00 001130'
 1219 001127' 260 17 0 00 012360'
 1220
 1221 001130' 271 03 0 00 000001
 1222 001131' 306 10 0 02 000000
 1223 001132' 254 00 0 00 001134'
 1224 001133' 260 17 0 00 012400'
 1225
 1226
 1227
 1228 001134' 312 03 0 00 011510'
 1229 001135' 254 00 0 00 001117'
 1230

;RECEIVE 3RD RESPONSE MESSAGE

T12A0: SETZM EXPSTA
 SETOM CNTFLG
 MOVEI S1,2*^D256
 MOVEM S1,EXPACT
 \$CALL RCVRM
 JRST T12AP
 ERROR1 (-1,EXPRSP,RCVRSP,Unexpected generate message response opcode,actual count=2
 ,NNN)
 JRST T12AU
 ERROR1 (-1,EXPSTA,RCVSTA,Unexpected generate message response status,actual count=2
 ,NNN)
 JRST T12AU
 ERROR1 (-1,EXPACT,RCVACT,Unexpected generate message response count,actual count=2,
 NNN)
 JRST T12AU
 ERROR1 (1,NUMREF,RNVNUM,Unexpected generate message response reference number,actua
 l count=2,NNN)
 JRST T12AU

;TEST DATA PAIR = 000,000/001,000

T12AP: SETZM T1 ; T1=BYTE COUNTER WITHIN BUFFER.
 MOVE S1,RETBUF ; S1=ADRS OF BUFFER.
 MOVE T4,[POINT 8,3(S1),15] ; T4=POINTER TO DATA.
 CLEARB P2,P3 ; CLEAR COMPARE REGS.
 JRST T12AR

;INCREMENT COMPARE REGISTERS.

T12AQ: ADDI P3,1 ; INCREMENT L/S BYTE OF BYTEPAIR.
 TRZE P3,400 ; TEST IF INCREMENT M/S BYTE NECESSARY.
 ADDI P2,1 ; INCREMENT M/S BYTE.

;COMPARE DATA BYTES.

T12AR: ADDI T1,1 ; INCR PRESENT BYTE COUNTER.
 ILDB T2,T4 ; T2=L/S BYTE DATA.
 ILDB S2,T4 ; S2=M/S BYTE DATA.
 CAIN P3,0(T2) ; COMPARE L/S BYTE.
 JRST T12AS
 ERROR1 (1,P3,T2,Data compare error on least significant byte,,NNNBYT)

T12AS: ADDI T1,1 ; INCR PRESENT BYTE COUNTER.
 CAIN P2,0(S2) ; COMPARE
 JRST T12AT ; EXIT
 ERROR1 (1,P2,S2,Data compare error on most significant byte,,NNNBYT)

;TEST IF DATA COMPARE COMPLETED.

T12AT: CAME T1,ITERCT ; TEST BYTE COUNT = DONE.
 JRST T12AQ ; BYTE COUNT NOT DONE,CONTINUE TESTING.

DFCIBM CTP Tests 01-80 version 2(20)
DFCIBM MAC 22-Aug-85 20:30

MACRO %53B(1242) 17:38 27-Aug-85 Page 27-1
TST12 Message Test - 4 bytes each of 4 data types (repeat=3)

SEQ 0366

1231
1232
1233 001136' 260 17 0 00 001067*

;Requeue the buffer AFTER THE 3RD MESSAGE RECEIVED AND TESTED.

T12AU: \$CALL REQUEM

1234
 1235
 1236
 1237
 1238
 1239
 1240 001137' 201 01 0 00 000000
 1241 001140' 201 02 0 00 012772'
 1242 001141' 260 17 0 00 011372'
 1243
 1244
 1245
 1246 001142' 201 01 0 00 000004
 1247 001143' 202 01 0 00 000734*
 1248 001144' 202 01 0 00 000735*
 1249 001145' 202 01 0 00 011510'
 1250
 1251 001146' 201 01 0 00 000000
 1252 001147' 202 01 0 00 000740*
 1253 001150' 202 01 0 00 000741*
 1254
 1255 001151' 201 01 0 00 000002
 1256 001152' 202 01 0 00 000743*
 1257
 1258 001153' 201 01 0 00 000002
 1259 001154' 202 01 0 00 000745*
 1260
 1261 001155' 260 17 0 00 000746*
 1262
 1263
 1264
 1265 001156' 260 17 0 00 011111'
 1266 001157' 254 00 0 00 001166*
 1267
 1268 001160' 260 17 0 00 012437'
 1269 001161' 254 00 0 00 001520*
 1270 001162' 260 17 0 00 012137'
 1271 001163' 254 00 0 00 001520*
 1272 001164' 260 17 0 00 012163*
 1273 001165' 254 00 0 00 001520*

```

;*****
;* TST12B - MESSAGE TEST. THIS TEST WILL SEND A MESSAGE REQUEST OF
; DATA=RESPONDER NODE, LENGTH=4, DELAY=0, REPEAT COUNT=2.
;*****

TST12B: MOVEI    S1,0                ;Load zero for no mesasge
        MOVEI    S2,[ASCIZ/Message Test (4 bytes of responder node repeated 3 times)/]
        $CALL    TRNODE              ;Copy message

; generate message ctp packet

        MOVEI    S1,4
        MOVEM    S1,OPCODE           ; OPCODE = GEN MESSAGE.
        MOVEM    S1,GENLEN           ; GEN LENGTH=4.
        MOVEM    S1,ITERCT          ; ITERCT=4.

        MOVEI    S1,0
        MOVEM    S1,DELAY            ; DELAY = 0
        MOVEM    S1,GNCNST           ; GENCONST=0.

        MOVEI    S1,2
        MOVEM    S1,RCOUNT           ; REPEAT COUNT = 2.

        MOVEI    S1,2
        MOVEM    S1,GENFUN           ; GENFUNCT =2/GENFNODE

        $CALL    BLDCTP

;TRANSMIT REQUEST MESSAGE

        $CALL    XMTREQ              ;Transmit request
        JRST     T12BA               ;OK
        ERROR1    (,,, <Timed out waiting for event pending after transmitting generate message
>,,NNN)
        JRST     TST12X
        ERROR1    (,,, <No send complete event after transmitting generate message>,,NNN)
        JRST     TST12X
        ERROR1    (,,, <Timed out waiting for generate message response available flag>,,NNN)
        JRST     TST12X

```

```

1274
1275
1276
1277 001166' 402 00 0 00 011430'
1278 001167' 402 00 0 00 011423'
1279 001170' 476 00 0 00 011422'
1280 001171' 260 17 0 00 011043'
1281 001172' 254 00 0 00 001203'
1282
1283 001173' 260 17 0 00 012536'
1284 001174' 254 00 0 00 001214'
1285
1286 001175' 260 17 0 00 012736'
1287 001176' 254 00 0 00 001214'
1288
1289 001177' 260 17 0 00 012563'
1290 001200' 254 00 0 00 001214'
1291
1292 001201' 260 17 0 00 013006'
1293 001202' 254 00 0 00 001214'
1294
1295
1296
1297
1298 001203' 402 00 0 00 000003
1299 001204' 200 01 0 00 001113*
1300 001205' 200 02 0 00 012314'
1301
1302 001206' 271 03 0 00 000001
1303 001207' 134 06 0 00 000002
1304 001210' 312 06 0 00 000442*
1305 001211' 260 17 0 00 013015'
1306
1307
1308
1309 001212' 312 03 0 00 011510'
1310 001213' 254 00 0 00 001206'
1311
1312
1313
1314 001214' 260 17 0 00 001136*
1315
1316
1317
1318 001215' 260 17 0 00 011330'
1319 001216' 254 00 0 00 001221'
1320 001217' 260 17 0 00 012613'
1321 001220' 254 00 0 00 001520'
  
```

;RECEIVE 1ST RESPONSE MESSAGE

```

T12BA: SETZM EXPSTA
        SETZM EXPACT
        SETOM CNIFLG
        $CALL RCVRM
        JRST T12BB
        ERROR1 (-1,EXPRSP,RCVRSP,Unexpected generate message response opcode,actual count=0
,NNN)
        JRST T12BD
        ERROR1 (-1,EXPSTA,RCVSTA,Unexpected generate message response status,actual count=0
,NNN)
        JRST T12BD
        ERROR1 (-1,EXPACT,RCVACT,Unexpected generate message response count,actual count=0,
NNN)
        JRST T12BD
        ERROR1 (1,NUMREF,RNVNUM,Unexpected generate message response reference number,actua
l count=0,NNN)
        JRST T12BD
  
```

;TEST DATA FIELD
 ;S1=DATA HOLDING REG,S2=DATA POINTER REG,T1=BYTE COUNTER

```

T12BB: SETZM T1 ; T1=BYTE COUNTER OF DATA READ BACK.
        MOVE S1,RETBUF
        MOVE S2,[POINT 8,3(S1),15] ; 1ST DATA BYTE TO BE EXAMINED.

T12BC: ADDI T1,1 ; INCR BYTE COUNT.
        ILDB T4,S2 ; LOAD DATA BYTE FROM BUFFER
        CAME T4,NODEN ; COMPARE TO NODE NUMBER.
        ERROR1 (1,NODEN,T4,Data compare error,,NNNBYT)
  
```

;TEST IF ALL BYTES DONE.

```

        CAME T1,ITERCT ; TEST IF ALL BUFFER TESTED.
        JRST T12BC ; CONTINUE IN DATA COMPARE LOOP.
  
```

;Requeue the buffer AFTER 1ST MESSAGE RECEIVED AND TESTED.

T12BD: \$CALL REQUEM

;WAIT FOR 2ND MESSAGE AVAILABLE

```

        $CALL MSGAVL ; WAIT FOR MESSAGE AVAILABLE FLAG
        JRST T12BE ; NO FAILURE,CONTINUE
        ERROR1 (,Time out waiting for generate message response,,NNN)
        JRST T12X
  
```


1322
 1323
 1324
 1325 001221' 402 00 0 00 011430'
 1326 001222' 476 00 0 00 011422'
 1327 001223' 201 01 0 00 000400
 1328 001224' 202 01 0 00 011423'
 1329 001225' 260 17 0 00 011043'
 1330 001226' 254 00 0 00 001237'
 1331
 1332 001227' 260 17 0 00 012625'
 1333 001230' 254 00 0 00 001250'
 1334
 1335 001231' 260 17 0 00 012634'
 1336 001232' 254 00 0 00 001250'
 1337
 1338 001233' 260 17 0 00 012643'
 1339 001234' 254 00 0 00 001250'
 1340
 1341 001235' 260 17 0 00 013024'
 1342 001236' 254 00 0 00 001250'
 1343
 1344
 1345
 1346
 1347 001237' 402 00 0 00 000003
 1348 001240' 200 01 0 00 001204*
 1349 001241' 200 02 0 00 012314'
 1350
 1351 001242' 271 03 0 00 000001
 1352 001243' 15 06 0 00 000002
 1353 001244' 312 06 0 00 001210*
 1354 001245' 260 17 0 00 013033'
 1355
 1356
 1357
 1358 001246' 312 03 0 00 011510'
 1359 001247' 254 00 0 00 001242'
 1360
 1361
 1362
 1363 001250' 260 17 0 00 001214*
 1364
 1365
 1366
 1367 001251' 260 17 0 00 011330'
 1368 001252' 254 00 0 00 001255'
 1369 001253' 260 17 0 00 012613'
 1370 001254' 254 00 0 00 001520'

;RECEIVE 2ND RESPONSE MESSAGE

T12BE: SETZM EXPSTA
 SETOM CNTFLG
 MOVEI S1,1*^D256
 MOVEM S1,EXPACT
 \$CALL RCVRM
 JRST T12BF
 ERROR1 (-1,EXPRSP,RCVRSP,Unexpected generate message response opcode,actual count=1
 ,NNN)
 JRST T12BH
 ERROR1 (-1,EXPSTA,RCVSTA,Unexpected generate message response status,actual count=1
 ,NNN)
 JRST T12BH
 ERROR1 (-1,EXPACT,RCVACT,Unexpected generate message response count,actual count=1,
 NNN)
 JRST T12BH
 ERROR1 (1,NUMREF,RNVNUM,Unexpected generate message response reference number,actua
 l count=1,NNN)
 JRST T12BH

;TEST DATA FIELD

;S1=DATA HOLDING REG,S2=DATA POINTER REG,T1=BYTE COUNTER

T12BF: SETZM T1 ; T1=BYTE COUNTER OF DATA READ BACK.
 MOVE S1,RETBUR
 MOVE S2,[POINT 8,3(S1),15] ; 1ST DATA BYTE TO BE EXAMINED.
 T12BG: ADDI T1,1 ; INCR BYTE COUNT.
 ILDB T4,S2 ; LOAD DATA BYTE FROM BUFFER
 CAME T4,NODEN ; COMPARE TO NODE NUMBER.
 ERROR1 (1,NODEN,T4,Data compare error,,NNNBYT)

;TEST IF ALL BYTES DONE.

CAME T1,ITERCT ; TEST IF ALL BUFFER TESTED.
 JRST T12BG ; CONTINUE IN DATA COMPARE LOOP.

;Requeue the buffer after 2nd message received and tested.

T12BH: \$CALL REQUEM

;WAIT FOR 3RD MESSAGE AVAILABLE

\$CALL MSGAVL ; WAIT FOR MESSAGE AVAILABLE FLAG
 JRST T12BI ; NO FAILURE,CONTINUE
 ERROR1 (,Time out waiting for generate message response,,NNN)
 JRST T12X

1371
 1372
 1373
 1374 001255' 402 00 0 00 011430'
 1375 001256' 476 00 0 00 011422'
 1376 001257' 201 01 0 00 001000'
 1377 001260' 202 01 0 00 011423'
 1378 001261' 260 17 0 00 011043'
 1379 001262' 254 00 0 00 001273'
 1380
 1381 001263' 260 17 0 00 012664'
 1382 001264' 254 00 0 00 001304'
 1383
 1384 001265' 260 17 0 00 012673'
 1385 001266' 254 00 0 00 001304'
 1386
 1387 001267' 260 17 0 00 012702'
 1388 001270' 254 00 0 00 001304'
 1389
 1390 001271' 260 17 0 00 013042'
 1391 001272' 254 00 0 00 001304'
 1392
 1393
 1394
 1395
 1396 001273' 402 00 0 00 000003'
 1397 001274' 200 01 0 00 001240*
 1398 001275' 200 02 0 00 012314'
 1399
 1400 001276' 271 03 0 00 000001'
 1401 001277' 13 06 0 00 000002'
 1402 001300' 312 06 0 00 001244*
 1403 001301' 260 17 0 00 013051'
 1404
 1405
 1406
 1407 001302' 312 03 0 00 011510'
 1408 001303' 254 00 0 00 001276'
 1409
 1410
 1411
 1412 001304' 260 17 0 00 001250*

;RECEIVE 3RD RESPONSE MESSAGE

T12BI: SETZM EXPSTA
 SETOM CNTFLG
 MOVEI S1,2*^D256
 MOVEM S1,EXPACT
 \$CALL RCVRM
 JRST T12BJ
 ERROR1 (-1,EXPRSP,RCVRSP,Unexpected generate message response opcode,actual count=2
 ,NNN)
 JRST T12BL
 ERROR1 (-1,EXPSTA,RCVSTA,Unexpected generate message response status,actual count=2
 ,NNN)
 JRST T12BL
 ERROR1 (-1,EXPACT,RCVACT,Unexpected generate message response count,actual count=2,
 NNN)
 JRST T12BL
 ERROR1 (1,NUMREF,RNVNUM,Unexpected generate message response reference number,actua
 l count=2,NNN)
 JRST T12BL

;TEST DATA FIELD

;S1=DATA HOLDING REG,S2=DATA POINTER REG,T1=BYTE COUNTER

T12BJ: SETZM T1 ; T1=BYTE COUNTER OF DATA READ BACK.
 MOVE S1,RETBUF
 MOVE S2,[POINT 8,3(S1),15] ; 1ST DATA BYTE TO BE EXAMINED.
 T12BK: ADDI T1,1 ; INCR BYTE COUNT.
 ILDB T4,S2 ; LOAD DATA BYTE FROM BUFFER
 CAME T4,NODEN ; COMPARE TO NODE NUMBER.
 ERROR1 (1,NODEN,T4,Data compare error,,NNNBYT)

;TEST IF ALL BYTES DONE.

CAME T1,ITERCT ; TEST IF ALL BUFFER TESTED.
 JRST T12BK ; CONTINUE IN DATA COMPARE LOOP.

;Requeue the buffer after 3rd message received and tested.

T12BL: \$CALL REQUEM

1413
 1414
 1415
 1416
 1417
 1418
 1419 001305' 201 01 0 00 000000
 1420 001306' 201 02 0 00 013060'
 1421 001307' 260 17 0 00 011372'
 1422
 1423
 1424
 1425 001310' 201 01 0 00 000004
 1426 001311' 202 01 0 00 001143*
 1427 001312' 202 01 0 00 001144*
 1428 001313' 202 01 0 00 011510'
 1429
 1430 001314' 201 01 0 00 000000
 1431 001315' 202 01 0 00 001147*
 1432 001316' 202 01 0 00 001150*
 1433
 1434 001317' 201 01 0 00 000002
 1435 001320' 202 01 0 00 001152*
 1436
 1437 001321' 201 01 0 00 000003
 1438 001322' 202 01 0 00 001154*
 1439
 1440 001323' 200 01 0 00 012474'
 1441 001324' 202 01 0 00 011433'
 1442 001325' 201 01 0 00 000004
 1443 001326' 202 01 0 00 011432'
 1444
 1445 001327' 260 17 0 00 001155*
 1446
 1447
 1448
 1449
 1450 001330' 260 17 0 00 011111'
 1451 001331' 254 00 0 00 001340'
 1452
 1453 001332' 260 17 0 00 012437'
 1454 001333' 254 00 0 00 001520'
 1455 001334' 260 17 0 00 012137'
 1456 001335' 254 00 0 00 001520'
 1457 001336' 260 17 0 00 012163'
 1458 001337' 254 00 0 00 001520'

 * TST12C - MESSAGE TEST. THIS TEST WILL SEND A MESSAGE REQUEST OF
 DATA=IMAGE DATA, LENGTH=4, DELAY=0, REPEAT COUNT=2.

TST12C: MOVEI S1,0 ;No message
 MOVEI S2,[ASCIZ/Message Test (4 bytes of Image Data)/]
 \$CALL TRNODE ;Set message up

;Build generate message ctp packet

MOVEI S1,4
 MOVEM S1,OPCODE ; OPCODE = GEN MESSAGE.
 MOVEM S1,GENLEN ; GEN LENGTH=4.
 MOVEM S1,ITERCT ; ITERCT=4.

 MOVEI S1,0
 MOVEM S1,DELAY ; DELAY = 0
 MOVEM S1,GNCNST ; GENCONST=0.

 MOVEI S1,2
 MOVEM S1,RCOUNT ; REPEAT COUNT = 2.

 MOVEI S1,3
 MOVEM S1,GENFUN ; GEFUNCT=3/GENFIMAGE

 MOVE S1,[777773757700] ; IMAGE=377,376,375,374.
 MOVEM S1,IMAGDT
 MOVEI S1,4
 MOVEM S1,IMGLEN

 \$CALL BLDCTP

;TRANSMIT REQUEST MESSAGE

\$CALL XMTREQ ;Transmit request
 JRST T12CA ;OK
 ERROR1 (,,, <Timed out waiting for event pending after transmitting generate message
 >,,NNN)
 JRST TST12X
 ERROR1 (,,, <No send complete event after transmitting generate message>,,NNN)
 JRST TST12X
 ERROR1 (,,, <Timed out waiting for generate message response available flag>,,NNN)
 JRST TST12X

1459
 1460
 1461
 1462 001340' 402 00 0 00 011423'
 1463 001341' 402 00 0 00 011430'
 1464 001342' 476 00 0 00 011422'
 1465 001343' 260 17 0 00 011043'
 1466 001344' 254 00 0 00 001355'
 1467
 1468 001345' 260 17 0 00 012536'
 1469 001346' 254 00 0 00 001401'
 1470
 1471 001347' 260 17 0 00 012736'
 1472 001350' 254 00 0 00 001401'
 1473
 1474 001351' 260 17 0 00 012563'
 1475 001352' 254 00 0 00 001401'
 1476
 1477 001353' 260 17 0 00 013070'
 1478 001354' 254 00 0 00 001401'
 1479
 1480
 1481
 1482
 1483
 1484
 1485
 1486
 1487
 1488 001355' 200 01 0 00 001274*
 1489 001356' 201 02 0 00 012314'
 1490 001357' 201 03 0 00 000000
 1491 001360' 201 05 0 00 011433'
 1492 001361' 275 05 0 00 000001
 1493 001362' 200 06 0 00 012504'
 1494 001363' 201 07 0 00 000000
 1495
 1496
 1497
 1498 001364' 134 04 0 00 000006
 1499 001365' 271 07 0 00 000001
 1500 001366' 312 07 0 00 011432'
 1501 001367' 254 00 0 00 001372'
 1502 001370' 201 07 0 00 000000
 1503 001371' 200 06 0 00 012504'

;RECEIVE 1ST RESPONSE MESSAGE

T12CA: SETZM EXPACT
 SETZM EXPSTA
 SETOM CNTFLG
 \$CALL RCVRM
 JRST T12CB
 ERROR1 (-1,EXPRSP,RCVRSP,Unexpected generate message response opcode,actual count=0
 ,NNN)
 JRST T12CE
 ERROR1 (-1,EXPSTA,RCVSTA,Unexpected generate message response status,actual count=0
 ,NNN)
 JRST T12CE
 ERROR1 (-1,EXPACT,RCVACT,Unexpected generate message response count,actual count=0,
 NNN)
 JRST T12CE
 ERROR1 (1,NUMREF,RNVNUM,Unexpected generate message response reference number,actua
 l count=0,NNN)
 JRST T12CE

;DATA COMPARISON ROUTINE FOR IMAGDT.

; REGISTER USAGE FOR BUFFER DATA:
 ; P2=DATA HOLDING REG OF BUFFER
 ; S2=BYTE POINTER FOR BUFFER
 ; REGISTER USAGE FOR IMAGE DATA:
 ; T2=DATA HOLDING REG OF IMAGE DATA
 ; T4=BYTE POINTER FOR IMAGE DATA
 S1=ADDRESS OF BUFFER
 T1=BYTE COUNTER OF BUFFER
 T3=ADDRESS OF IMAGD
 P1=BYTE COUNTER OF IMAGDT

T12CB: MOVE S1,RETBUF
 MOVE S2,[POINT 8,3(S1),15] ; 1ST DATA BYTE TO BE EXAMINED-1.
 MOVEI T1,0 ; RESET BUFFER BYTE COUNTER
 MOVEI T3,IMAGDT
 SUBI T3,1
 MOVE T4,[POINT 8,(T3),31] ; 1ST IMAGE BYTE TO BE EXAMINED-1.
 MOVEI P1,0 ; RESET IMAGE BYTE COUNTER.

;GET IMAGE DATA BYTE.

T12CC: ILDB T2,T4 ; T2 = IMAGE DATA BYTE.
 ADDI P1,1 ; INCR IMAGE DATA COUNT.
 CAME P1,IMGLEN ; TEST IF IMAGE LENGTH DONE.
 JRST T12CD ; NO, CONTINUE WITH CURRENT COMPARE.
 MOVEI P1,0 ; RESET CURRENT IMAGE COUNT.
 MOVE T4,[POINT 8,(T3),31] ; RESET IMAGE DATA POINTER.

1504
 1505
 1506
 1507 001372' 134 10 0 00 000002
 1508
 1509
 1510
 1511 001373' 302 10 0 04 000000
 1512 001374' 260 17 0 00 012505'
 1513
 1514
 1515
 1516 001375' 271 03 0 00 000001
 1517 001376' 311 03 0 00 011510'
 1518 001377' 254 00 0 00 001401'
 1519
 1520 001400' 254 00 0 00 001364'
 1521
 1522
 1523
 1524 001401' 260 17 0 00 001304*
 1525
 1526
 1527
 1528 001402' 260 17 0 00 011330'
 1529 001403' 254 00 0 00 001406'
 1530 001404' 260 17 0 00 012613'
 1531 001405' 254 00 0 00 001520'
 1532
 1533
 1534
 1535
 1536 001406' 402 00 0 00 011430'
 1537 001407' 476 00 0 00 011422'
 1538 001410' 201 01 0 00 000400
 1539 001411' 202 01 0 00 011423'
 1540 001412' 260 17 0 00 011043'
 1541
 1542 001413' 254 00 0 00 001424'
 1543
 1544 001414' 260 17 0 00 012625'
 1545 001415' 254 00 0 00 001450'
 1546
 1547 001416' 260 17 0 00 012634'
 1548 001417' 254 00 0 00 001450'
 1549
 1550 001420' 260 17 0 00 012643'
 1551 001421' 254 00 0 00 001450'
 1552
 1553 001422' 260 17 0 00 013077'
 1554 001423' 254 00 0 00 001450'

```
;GET BUFFER DATA BYTE.
T12CD: ILDB P2,S2 ; P2=BUFFER DATA BYTE
;COMPARE BUFFER DATA WITH IMAGE DATA.
CAIE P2,0(T2)
ERROR1 (1,T2,P2,Data compare error,,NNNBYT)
;COMPARE BUFFER BYTE COUNT WITH COMMAND BYTE COUNT.
ADDI T1,1 ; INCR PRESENT BYTE POINTER
CAML T1,ITERCT ; TEST IF PRESENT = ISSUED BYTES.
JRST T12CE ; JUMP IF ALL BUFFER TESTED.
JRST T12CC ; CONTINUE 'TIL ALL DATA BLOCK TESTED.
;Requeue the buffer AFTER 1ST MESSAGE RECEIVED AND TESTED.
T12CE: $CALL REQUEM
;WAIT FOR 2ND MESSAGE AVAILABLE
$CALL MSGAVL ; WAIT FOR MESSAGE AVAILABLE FLAG
JRST T12CF ; NO FAILURE,CONTINUE
ERROR1 (,Time out waiting for generate message response,,NNN)
JRST TST12X
;RECEIVE 2ND RESPONSE MESSAGE
;TEST RESPONSE MESSAGE OPCODE, STATUS,ACTCNT, AND REFERENCE NUMBER
T12CF: SETZM EXPSTA
SETOM CNTFLG
MOVEI S1,1*D256
MOVEM S1,EXPACT
$CALL RCVRM
JRST T12CG
ERROR1 (-1,EXPRSP,RCVRSP,Unexpected generate message response opcode,actual count=1,NNN)
JRST T12CJ
ERROR1 (-1,EXPSTA,RCVSTA,Unexpected generate message response status,actual count=1,NNN)
JRST T12CJ
ERROR1 (-1,EXPACT,RCVACT,Unexpected generate message response count,actual count=1,NNN)
JRST T12CJ
ERROR1 (1,NUMREF,RNVNUM,Unexpected generate message response reference number,actual count=1,NNN)
JRST T12CJ
```


1555
 1556
 1557
 1558
 1559
 1560
 1561
 1562
 1563
 1564 001424' 200 01 0 00 001355*
 1565 001425' 200 02 0 00 012314'
 1566 001426' 201 03 0 00 000000
 1567 001427' 201 05 0 00 011433'
 1568 001430' 275 05 0 00 000001
 1569 001431' 200 06 0 00 012504'
 1570 001432' 201 07 0 00 000000
 1571
 1572
 1573
 1574 001433' 134 04 0 00 000006
 1575 001434' 271 07 0 00 000001
 1576 001435' 312 07 0 00 011432'
 1577 001436' 254 00 0 00 001441'
 1578 001437' 201 07 0 00 000000
 1579 001440' 200 06 0 00 012504'
 1580
 1581
 1582
 1583 001441' 134 10 0 00 000002
 1584
 1585
 1586
 1587 001442' 302 10 0 04 000000
 1588 001443' 260 17 0 00 012505'
 1589
 1590
 1591
 1592 001444' 271 03 0 00 000001
 1593 001445' 311 03 0 00 011510'
 1594 001446' 254 00 0 00 001450'
 1595
 1596 001447' 254 00 0 00 001433'
 1597
 1598
 1599
 1600 001450' 260 17 0 00 001401*

```

;DATA COMPARISON ROUTINE FOR IMAGDT.
; REGISTER USAGE FOR BUFFER DATA:
; P2=DATA HOLDING REG OF BUFFER          S1=ADDRESS OF BUFFER
; S2=BYTE POINTER FOR BUFFER             T1=BYTE COUNTER OF BUFFER
; REGISTER USAGE FOR IMAGE DATA:
; T2=DATA HOLDING REG OF IMAGE DATA      T3=ADDRESS OF IMAGDT
; T4=BYTE POINTER FOR IMAGE DATA         P1=BYTE COUNTER OF IMAGDT

T12CG:  MOVE    S1,RETBUF
        MOVE    S2,[POINT 8,3(S1),15]    ; 1ST DATA BYTE TO BE EXAMINED-1.
        MOVEI   T1,0                      ; RESET BUFFER BYTE COUNTER
        MOVEI   T3,IMAGDT
        SUBI    T3,1
        MOVE    T4,[POINT 8,(T3),31]      ; 1ST IMAGE BYTE TO BE EXAMINED-1.
        MOVEI   P1,0                      ; RESET IMAGE BYTE COUNTER.

;GET IMAGE DATA BYTE.
T12CH:  ILDB    T2,T4                      ; T2 = IMAGE DATA BYTE.
        ADDI    P1,1                      ; INCR IMAGE DATA COUNT.
        CAML    P1,IMGLN                  ; TEST IF IMAGE LENGTH DONE.
        JRST    T12CI                     ; NO, CONTINUE WITH CURRENT COMPARE.
        MOVEI   P1,0                      ; RESET CURRENT IMAGE COUNT.
        MOVE    T4,[POINT 8,(T3),31]      ; RESET IMAGE DATA POINTER.

;GET BUFFER DATA BYTE.
T12CI:  ILDB    P2,S2                      ; P2=BUFFER DATA BYTE

;COMPARE BUFFER DATA WITH IMAGE DATA.
        CAIE    P2,0(T2)
        ERROR1  (1,T2,P2,Data compare error,,NNNBYT)

;COMPARE BUFFER BYTE COUNT WITH COMMAND BYTE COUNT.
        ADDI    T1,1                      ; INCR PRESENT BYTE POINTER
        CAML    T1,ITERCT                 ; TEST IF PRESENT = ISSUED BYTES.
        JRST    T12CJ                     ; JUMP IF ALL BUFFER TESTED.

        JRST    T12CH                     ; CONTINUE 'TIL ALL DATA BLOCK TESTED.

;Requeue the buffer after 2nd message received and tested.
T12CJ:  $CALL   REQUEM

```


1601
 1602
 1603
 1604 001451' 260 17 0 00 011330'
 1605 001452' 254 00 0 00 001455'
 1606 001453' 260 17 0 00 012613'
 1607 001454' 254 00 0 00 001520'
 1608
 1609
 1610
 1611
 1612 001455' 402 00 0 00 011430'
 1613 001456' 476 00 0 00 011422'
 1614 001457' 201 01 0 00 001000'
 1615 001460' 202 01 0 00 011423'
 1616 001461' 260 17 0 00 011043'
 1617
 1618 001462' 254 00 0 00 001473'
 1619
 1620 001463' 260 17 0 00 012664'
 1621 001464' 254 00 0 00 001517'
 1622
 1623 001465' 260 17 0 00 012673'
 1624 001466' 254 00 0 00 001517'
 1625
 1626 001467' 260 17 0 00 012702'
 1627 001470' 254 00 0 00 001517'
 1628
 1629 001471' 260 17 0 00 013106'
 1630 001472' 254 00 0 00 001517'
 1631
 1632
 1633
 1634
 1635
 1636
 1637
 1638
 1639
 1640 001473' 200 01 0 00 001424*
 1641 001474' 200 02 0 00 012314'
 1642 001475' 201 03 0 00 000000'
 1643 001476' 201 05 0 00 011433'
 1644 001477' 275 05 0 00 000001'
 1645 001500' 200 06 0 00 012504'
 1646 001501' 201 07 0 00 000000'

;WAIT FOR 3RD MESSAGE AVAILABLE

SCALL MSGAVL ; WAIT FOR MESSAGE AVAILABLE FLAG
 JRST T12CK ; NO FAILURE,CONTINUE
 ERROR1 (,Time out waiting for generate message response,,NNN)
 JRST T12X

;RECEIVE 3RD RESPONSE MESSAGE
 ;TEST RESPONSE MESSAGE OPCODE, STATUS,ACTCNT, AND REFERENCE NUMBER

T12CK: SETZM EXPSTA
 SETOM CNTFLG
 MOVEI S1,2*^D256
 MOVEM S1,EXPACT
 SCALL RCVRM
 JRST T12CL
 ERROR1 (-1,EXPRSP,RCVRSP,Unexpected generate message response opcode,actual count=2,
 ,NNN)
 JRST T12CO
 ERROR1 (-1,EXPSTA,RCVSTA,Unexpected generate message response status,actual count=2,
 ,NNN)
 JRST T12CO
 ERROR1 (-1,EXPACT,RCVACT,Unexpected generate message response count,actual count=2,
 ,NNN)
 JRST T12CO
 ERROR1 (1,NUMREF,RNVNUM,Unexpected generate message response reference number,actual
 count=2,,NNN)
 JRST T12CO

;DATA COMPARISON ROUTINE FOR IMAGDT.
 ; REGISTER USAGE FOR BUFFER DATA:
 ; P2=DATA HOLDING REG OF BUFFER
 ; S2=BYTE POINTER FOR BUFFER
 ; REGISTER USAGE FOR IMAGE DATA:
 ; T2=DATA HOLDING REG OF IMAGE DATA
 ; T4=BYTE POINTER FOR IMAGE DATA

S1=ADDRESS OF BUFFER
 T1=BYTE COUNTER OF BUFFER
 T3=ADDRESS OF IMAGDT
 P1=BYTE COUNTER OF IMAGDT

T12CL: MOVE S1,RETBUF
 MOVE S2,[POINT 8,3(S1),15] ; 1ST DATA BYTE TO BE EXAMINED-1.
 MOVEI T1,0 ; RESET BUFFER BYTE COUNTER
 MOVEI T3,IMAGDT
 SUBI T3,1
 MOVE T4,[POINT 8,(T3),31] ; 1ST IMAGE BYTE TO BE EXAMINED-1.
 MOVEI P1,0 ; RESET IMAGE BYTE COUNTER.

1647
 1648
 1649
 1650 001502' 134 04 0 00 000006
 1651 001503' 271 07 0 00 000001
 1652 001504' 312 07 0 00 011432'
 1653 001505' 254 00 0 00 001510'
 1654 001506' 201 07 0 00 000000
 1655 001507' 200 06 0 00 012504'
 1656
 1657
 1658
 1659 001510' 134 10 0 00 000002
 1660
 1661
 1662
 1663 001511' 302 10 0 04 000000
 1664 001512' 260 17 0 00 012505'
 1665
 1666
 1667
 1668 001513' 271 03 0 00 000001
 1669 001514' 311 03 0 00 011510'
 1670 001515' 254 00 0 00 001517'
 1671 001516' 254 00 0 00 001502'
 1672
 1673
 1674
 1675 001517' 260 17 0 00 001450*
 1676
 1677
 1678
 1679 001520' 260 17 0 00 000535*
 1680
 1681
 1682
 1683 001521' 260 17 0 00 011306'
 1684 001522' 254 00 0 00 001525'
 1685 001523' 260 17 0 00 011650'
 1686 001524' 254 00 0 00 001527'
 1687
 1688
 1689
 1690 001525' 201 01 0 00 000764
 1691 001526' 260 17 0 00 000543*
 1692 001527' 263 17 0 00 000000

;GET IMAGE DATA BYTE.

T12CM: ILDB T2,T4 ; T2 = IMAGE DATA BYTE.
 ADDI P1,1 ; INCR IMAGE DATA COUNT.
 CAME P1,IMGLEN ; TEST IF IMAGE LENGTH DONE.
 JRST T12CN ; NO, CONTINUE WITH CURRENT COMPARE.
 MOVEI P1,0 ; RESET CURRENT IMAGE COUNT.
 MOVE T4,[POINT 8,(T3),31] ; RESET IMAGE DATA POINTER.

;GET BUFFER DATA BYTE.

T12CN: ILDB P2,S2 ; P2=BUFFER DATA BYTE

;COMPARE BUFFER DATA WITH IMAGE DATA.

CAIE P2,0(T2)
 ERROR1 (1,T2,P2,Data compare error,,NNNBYT)

;COMPARE BUFFER BYTE COUNT WITH COMMAND BYTE COUNT.

ADDI T1,1 ; INCR PRESENT BYTE POINTER
 CAML T1,ITERCT ; TEST IF PRESENT = ISSUED BYTES.
 JRST T12CO ; JUMP IF ALL BUFFER TESTED.
 JRST T12CM ; CONTINUE 'TIL ALL DATA BLOCK TESTED.

;Requeue the buffer AFTER 3RD MESSAGE RECEIVED AND TESTED.

T12CO: \$CALL REQUEM

;Disconnect with selected node

TST12X: \$CALL DODIS ;Disconnect with the selected node

;Wait for connection closed

\$CALL CONCLD
 JRST TST12Y
 ERROR1 (,,Connection closed not in disconnect status,,NNN)
 JRST TST12Z

;Wait

TST12Y: MOVEI S1,^D500
 \$CALL WAITX
 TST12Z: \$RET

DFCIBM CTP Tests 01-80 version 2(20)
 DFCIBM MAC 22-Aug-85 20:30

MACRO %53B(1242) 17:38 27-Aug-85 Page 38
 TST13 - Message Test - 4 bytes of 4 data types (delay of 2 sec)

SEQ 0377

1693
 1694
 1695
 1696
 1697
 1698
 1699
 1700 001530' 201 01 0 00 013115'
 1701 001531' 201 02 0 00 013117'
 1702 001532' 260 17 0 00 011372'
 1703
 1704
 1705
 1706 001533' 260 17 0 00 011134'
 1707 001534' 254 00 0 00 001543'
 1708
 1709 001535' 260 17 0 00 011562'
 1710 001536' 254 00 0 00 002036'
 1711
 1712 001537' 260 17 0 00 011607'
 1713 001540' 254 00 0 00 002036'
 1714 001541' 260 17 0 00 011630'
 1715 001542' 254 00 0 00 002036'
 1716
 1717
 1718
 1719 001543' 201 01 0 00 000004
 1720 001544' 202 01 0 00 001311*
 1721 001545' 202 01 0 00 001312*
 1722 001546' 202 01 0 00 011510'
 1723
 1724 001547' 201 01 0 00 000000
 1725 001550' 202 01 0 00 001322*
 1726 001551' 202 01 0 00 001320*
 1727
 1728 001552' 201 01 0 00 000004
 1729 001553' 202 01 0 00 001315*
 1730
 1731 001554' 201 01 0 00 000375
 1732 001555' 202 01 0 00 001316*
 1733
 1734 001556' 260 17 0 00 001327*
 1735
 1736
 1737
 1738 001557' 260 17 0 00 011111'
 1739 001560' 254 00 0 00 001567'
 1740
 1741 001561' 260 17 0 00 012437'
 1742 001562' 254 00 0 00 002031'
 1743 001563' 260 17 0 00 012137'
 1744 001564' 254 00 0 00 002031'
 1745 001565' 260 17 0 00 012163'
 1746 001566' 254 00 0 00 002031'

SUBTTL TST13 - Message Test - 4 bytes of 4 data types (delay of 2 sec)

 * TST13 - MESSAGE TEST. THIS TEST WILL SEND A MESSAGE REQUEST OF
 DATA=375, LENGTH=4, DELAY=4(2 SECONDS), REPEAT COUNT=0.

TST13: MOVEI S1,[ASCIZ/TST13/] ;Load test name
 MOVEI S2,[ASCIZ/Message Test (4 bytes of 375 with 2 second delay)/]
 \$CALL TRNODE ;Print node number if trace switch set

;Connect with selected node and verify connection=open.

\$CALL CONNCT ;Connect with the selected node
 JRST T13A ;Connect OK
 ERROR1 (,,,Timed out waiting for event pending flag during connection attempt,,NNN)

JRST TST13Y
 ERROR1 (,,,Event pending was not connection accepted during connection attempt,,NNN)

JRST TST13Y
 ERROR1 (,,,Connection not open after connection accepted,,NNN)
 JRST TST13Y

;Build generate message CTP packet

T13A: MOVEI S1,4
 MOVEM S1,OPCODE ; OPCODE = GEN MESSAGE.
 MOVEM S1,GENLEN ; GEN LENGTH=4.
 MOVEM S1,ITERCT ; ITERCT=4.
 MOVEI S1,0
 MOVEM S1,GENFUN ; GENFUNCT =0/GENFILL.
 MOVEM S1,RCOUNT ; REPEAT COUNT = 0.
 MOVEI S1,4
 MOVEM S1,DELAY ; DELAY = 4 (2 SECONDS).
 MOVEI S1,375
 MOVEM S1,GENCONST ; GENCONST=375.
 \$CALL BLDCTP

;TRANSMIT REQUEST MESSAGE

\$CALL XMTREQ ;Transmit request
 JRST T13B ;OK
 ERROR1 (,,,<Timed out waiting for event pending after transmitting generate message
 >,,NNN)
 JRST TST13X
 ERROR1 (,,,<No send complete event after transmitting generate message>,,NNN)
 JRST TST13X
 ERROR1 (,,,<Timed out waiting for generate message response available flag>,,NNN)
 JRST TST13X

1747
 1748
 1749
 1750 001567' 402 00 0 00 011430'
 1751 001570' 402 00 0 00 011423'
 1752 001571' 476 00 0 00 011422'
 1753 001572' 260 17 0 00 011043'
 1754
 1755 001573' 254 00 0 00 001604'
 1756 001574' 260 17 0 00 012203'
 1757 001575' 254 00 0 00 001615'
 1758 001576' 260 17 0 00 012223'
 1759 001577' 254 00 0 00 001615'
 1760 001600' 260 17 0 00 012243'
 1761 001601' 254 00 0 00 001615'
 1762
 1763 001602' 260 17 0 00 013131'
 1764 001603' 254 00 0 00 001615'
 1765
 1766
 1767
 1768 001604' 402 00 0 00 000003'
 1769 001605' 200 01 0 00 001473*
 1770 001606' 200 02 0 00 012314'
 1771
 1772 001607' 134 06 0 00 000002'
 1773 001610' 312 06 0 00 001555*
 1774 001611' 260 17 0 00 013140'
 1775
 1776
 1777
 1778 001612' 271 03 0 00 000001'
 1779 001613' 312 03 0 00 011510'
 1780 001614' 254 00 0 00 001607'
 1781
 1782
 1783
 1784 001615' 260 17 0 00 001517*

;RECEIVE RESPONSE MESSAGE

T13B: SETZM EXPSTA
 SETZM EXPACT
 SETOM CNTFLG
 \$CALL RCVRM
 JRST T13C
 ERROR1 (-1,EXPRSP,RCVRSP,Unexpected generate message response opcode,,NNN)
 JRST T13F
 ERROR1 (-1,EXPSTA,RCVSTA,Unexpected generate message response status,,NNN)
 JRST T13F
 ERROR1 (-1,EXPACT,RCVACT,Unexpected generate message response count,,NNN)
 JRST T13F
 ERROR1 (1,NUMREF,RNVNUM,Unexpected generate message response reference number,,NNN)
 JRST T13F

;TEST DATA FIELD FOR CORRECT DATA.

T13C: SETZM T1
 MOVE S1,RETBUF
 MOVE S2,[POINT 8,3(S1),15] ; SET POINTER IN S2.
 T13D: ILDB T4,S2 ; DATA INT T4.
 CAME T4,GNCNST ; COMPARE DATA TO EXPECTED.
 ERROR1 (1,GNCNST,T4,Data compare error,,NNNBYT)

;TEST IF DATA COMPARE COMPLETED.

T13E: ADDI T1,1
 CAME T1,ITERCT ; TEST BYTE COUNT = DONE.
 JRST T13D ; BYTE COUNT NOT DONE,CONTINUE TESTING.

;Requeue the buffer

T13F: \$CALL REQUEM

1785
 1786
 1787
 1788
 1789
 1790
 1791 001616' 201 01 0 00 000000
 1792 001617' 201 02 0 00 013147'
 1793 001620' 260 17 0 00 011372'
 1794
 1795
 1796
 1797 001621' 201 01 0 00 000004
 1798 001622' 202 01 0 00 001553*
 1799
 1800 001623' 201 01 0 00 000000
 1801 001624' 202 01 0 00 001610*
 1802
 1803 001625' 201 01 0 00 000001
 1804 001626' 202 01 0 00 001550*
 1805
 1806 001627' 260 17 0 00 001556*
 1807
 1808
 1809
 1810
 1811 001630' 260 17 0 00 011111'
 1812 001631' 254 00 0 00 001640'
 1813
 1814 001632' 260 17 0 00 012437'
 1815 001633' 254 00 0 00 002031'
 1816 001634' 260 17 0 00 012137'
 1817 001635' 254 00 0 00 002031'
 1818 001636' 260 17 0 00 012163'
 1819 001637' 254 00 0 00 002031'
 1820
 1821
 1822
 1823 001640' 402 00 0 00 011430'
 1824 001641' 402 00 0 00 011423'
 1825 001642' 476 00 0 00 011422'
 1826 001643' 260 17 0 00 011043'
 1827
 1828 001644' 254 00 0 00 001655'
 1829 001645' 260 17 0 00 012203'
 1830 001646' 254 00 0 00 001673'
 1831 001647' 260 17 0 00 012223'
 1832 001650' 254 00 0 00 001673'
 1833 001651' 260 17 0 00 012243'
 1834 001652' 254 00 0 00 001673'
 1835
 1836 001653' 260 17 0 00 013163'
 1837 001654' 254 00 0 00 001673'

 * TST13A - MESSAGE TEST. THIS TEST WILL SEND A MESSAGE REQUEST OF
 DATA=BYTEPAIR, LENGTH=4, DELAY=4(2 SECONDS), REPEAT COUNT=0.

TST13A: MOVEI S1,0 ;Load 0 for no message output
 MOVEI S2,CASCIZ/Message Test (4 bytes of byte pair with 2 second delay)/]
 \$CALL TRNODE ;Store that message please

;build generate message ctp packet

MOVEI S1,4
 MOVEM S1,DELAY ; DELAY = 4 (2 SECONDS).

MOVEI S1,0
 MOVEM S1,GENCNST ; GENCONST=0.

MOVEI S1,1
 MOVEM S1,GENFUN ; GENFUNCT =1/GENFBPAIR.

\$CALL BLDCTP

;TRANSMIT REQUEST MESSAGE

\$CALL XMTREQ ;Transmit request
 JRST TA13A ;OK
 ERROR1 (,,, <Timed out waiting for event pending after transmitting generate message
 >,,NNN)
 JRST TST13X
 ERROR1 (,,, <No send complete event after transmitting generate message>,,NNN)
 JRST TST13X
 ERROR1 (,,, <Timed out waiting for generate message response available flag>,,NNN)
 JRST TST13X

;RECEIVE RESPONSE MESSAGE

TA13A: SETZM EXPSTA
 SETZM EXPACT
 SETOM CNTFLG
 \$CALL RCVRM
 JRST TA13C
 ERROR1 (-1,EXPRSP,RCVRSP,Unexpected generate message response opcode,,NNN)
 JRST TA13F
 ERROR1 (-1,EXPSTA,RCVSTA,Unexpected generate message response status,,NNN)
 JRST TA13F
 ERROR1 (-1,EXPACT,RCVACT,Unexpected generate message response count,,NNN)
 JRST TA13F
 ERROR1 (1,NUMREF,RNVNUM,Unexpected generate message response reference number,,NNN)
 JRST TA13F

1838
 1839
 1840
 1841 001655' 402 00 0 00 000003
 1842 001656' 200 01 0 00 001605*
 1843 001657' 200 06 0 00 012314'
 1844 001660' 403 10 0 00 000011
 1845 001661' 254 00 0 00 001665'
 1846
 1847
 1848
 1849 001662' 271 11 0 00 000001
 1850 001663' 622 11 0 00 000400
 1851 001664' 271 10 0 00 000001
 1852
 1853
 1854
 1855 001665' 271 03 0 00 000001
 1856 001666' 134 04 0 00 000006
 1857 001667' 134 02 0 00 000006
 1858 001670' 306 11 0 04 000000
 1859 001671' 254 00 0 00 001673'
 1860 001672' 260 17 0 00 012360'
 1861
 1862 001673' 271 03 0 00 000001
 1863 001674' 306 10 0 02 000000
 1864 001675' 254 00 0 00 001677'
 1865 001676' 260 17 0 00 012400'
 1866
 1867
 1868
 1869 001677' 312 03 0 00 011510'
 1870 001700' 254 00 0 00 001662'
 1871
 1872
 1873
 1874 001701' 260 17 0 00 001615*

;TEST DATA PAIR = 000,000/001,000

TA13C: SETZM T1 ; T1=BYTE COUNTER WITHIN BUFFER.
 MOVE S1,RETBUF ; S1=ADRS OF BUFFER.
 MOVE T4,[POINT 8,3(S1),15] ; T4=POINTER TO DATA.
 CLEARB P2,P3 ; CLEAR COMPARE REGS.
 JRST TA13E

;INCREMENT COMPARE REGISTERS.

TA13D: ADDI P3,1 ; INCREMENT L/S BYTE OF BYTEPAIR.
 TRZE P3,400 ; TEST IF INCREMENT M/S BYTE NECESSARY.
 ADDI P2,1 ; INCREMENT M/S BYTE.

;COMPARE DATA BYTES.

TA13E: ADDI T1,1 ; INCR PRESENT BYTE COUNTER.
 ILDB T2,T4 ; T2=L/S BYTE DATA.
 ILDB S2,T4 ; S2=M/S BYTE DATA.
 CAIN P3,0(T2) ; COMPARE L/S BYTE.
 JRST TA13F
 ERROR1 (1,P3,T2,Data compare error on least significant byte,,NNNBYT)

TA13F: ADDI T1,1 ; INCR PRESENT BYTE COUNTER.
 CAIN P2,0(S2) ; COMPARE
 JRST TA13G ; EXIT
 ERROR1 (1,P2,S2,Data compare error on most significant byte,,NNNBYT)

;TEST IF DATA COMPARE COMPLETED.

TA13G: CAME T1,ITERCT ; TEST BYTE COUNT = DONE.
 JRST TA13D ; BYTE COUNT NOT DONE,CONTINUE TESTING.

;Requeue the buffer

TA13H: \$CALL REQUEM

```

1875
1876
1877
1878
1879
1880
1881 001702' 201 01 0 00 000000
1882 001703' 201 02 0 00 013172'
1883 001704' 260 17 0 00 011372'
1884
1885
1886
1887 001705' 201 01 0 00 000002
1888 001706' 202 01 0 00 001626*
1889
1890 001707' 260 17 0 00 001627*
1891
1892
1893
1894 001710' 260 17 0 00 011111'
1895 001711' 254 00 0 00 001720'
1896
1897 001712' 260 17 0 00 012437'
1898 001713' 254 00 0 00 002031'
1899 001714' 260 17 0 00 012137'
1900 001715' 254 00 0 00 002031'
1901 001716' 260 17 0 00 012163'
1902 001717' 254 00 0 00 002031'
1903
1904
1905
1906 001720' 402 00 0 00 011430'
1907 001721' 402 00 0 00 011423'
1908 001722' 476 00 0 00 011422'
1909 001723' 260 17 0 00 011043'
1910
1911 001724' 254 00 0 00 001735'
1912 001725' 260 17 0 00 012203'
1913 001726' 254 00 0 00 001743'
1914 001727' 260 17 0 00 012223'
1915 001730' 254 00 0 00 001743'
1916 001731' 260 17 0 00 012243'
1917 001732' 254 00 0 00 001743'
1918
1919 001733' 260 17 0 00 013207'
1920 001734' 254 00 0 00 001733'

```

```

;*****
;* TST13B - MESSAGE TEST. THIS TEST WILL SEND A MESSAGE REQUEST OF DATA=
;* RESPONDER NODE, LENGTH=4, DELAY=4(2SECONDS), REPEAT CNT=0.
;*****

TST13B: MOVEI    S1,0                ;Load 0 for no message output
        MOVEI    S2,ASCIZ/Message Test (4 bytes of Responder Node with 2 second delay)/]
        SCALL    TRNODE             ;Store that message please

;build generate message ctp packet

        movei    S1,2
        movem    S1,genfun          ; genfunct =2/genfNODE
        $call    bldctp

;TRANSMIT REQUEST MESSAGE

        $CALL    XMTREQ              ;Transmit request
        JRST     TB13A              ;OK
        ERROR1   (,,, <Timed out waiting for event pending after transmitting generate message
>,,NNN)
        JRST     TST13X
        ERROR1   (,,, <No send complete event after transmitting generate message>,,NNN)
        JRST     TST13X
        ERROR1   (,,, <Timed out waiting for generate message response available flag>,,NNN)
        JRST     TST13X

;RECEIVE RESPONSE MESSAGE

TB13A:   SETZM    EXPSTA
        SETZM    EXPACT
        SETOM    CNTFLG
        $CALL    RCVRM

        JRST     TB13C
        ERROR1   (-1,EXPRSP,RCVRSP,Unexpected generate message response opcode,,NNN)
        JRST     TB13E
        ERROR1   (-1,EXPSTA,RCVSTA,Unexpected generate message response status,,NNN)
        JRST     TB13E
        ERROR1   (-1,EXPACT,RCVACT,Unexpected generate message response count,,NNN)
        JRST     TB13E
        ERROR1   (1,NUMREF,RNVNUM,Unexpected generate message response reference number,,NNN)

        JRST     TB13E

```

1921
 1922
 1923
 1924 001735' 402 00 0 00 000003
 1925 001736' 200 01 0 00 001656*
 1926 001737' 200 02 0 00 012314'
 1927 001740' 134 06 0 00 000002
 1928 001741' 312 06 0 00 001300*
 1929 001742' 260 17 0 00 013216'
 1930
 1931
 1932
 1933 001743' 271 03 0 00 000001
 1934 001744' 312 03 0 00 011510'
 1935 001745' 254 00 0 00 001740'
 1936
 1937
 1938 001746' 260 17 0 00 001701*
 1939
 1940
 1941
 1942
 1943
 1944
 1945 001747' 201 01 0 00 000000
 1946 001750' 201 02 0 00 013225'
 1947 001751' 260 17 0 00 011372'
 1948
 1949
 1950
 1951 001752' 201 01 0 00 000004
 1952 001753' 202 01 0 00 011510'
 1953
 1954 001754' 201 01 0 00 000003
 1955 001755' 202 01 0 00 001706*
 1956
 1957 001756' 260 17 0 00 001707*
 1958
 1959
 1960
 1961 001757' 260 17 0 00 011111'
 1962 001760' 254 00 0 00 001767'
 1963
 1964 001761' 260 17 0 00 012437'
 1965 001762' 254 00 0 00 002031'
 1966 001763' 260 17 0 00 012137'
 1967 001764' 254 00 0 00 002031'
 1968 001765' 260 17 0 00 012163'
 1969 001766' 254 00 0 00 002031'

;TEST DATA FIELD FOR CORRECT DATA.

TB13C: SETZM T1
 MOVE S1,RETBUF
 MOVE S2,[POINT 8,3(S1),15] ; SET POINTER IN S2.
 TB13D: ILDB T4,S2 ; DATA INT T4
 CAME T4,NODEN ; COMPARE DATA TO EXPECTED.
 ERROR1 (1,NODEN,T4,Data compare error,,NNNBYT)

;TEST IF DATA COMPARE COMPLETED.

TB13E: ADDI T1,1 ; INCR BYTE COUNT.
 CAME T1,ITERCT ; TEST BYTE COUNT = DONE.
 JRST TB13D ; BYTE COUNT NOT DONE,CONTINUE TESTING.

;Requeue the buffer

TB13F: \$CALL REQUEM

 * TST13C - MESSAGE TEST. THIS TEST WILL SEND A MESSAGE REQUEST OF DATA=IMAGE
 DATA(374,375,376,377), LENGTH=4, DELAY=4(2SECONDS), REPEAT CNT=0.

TST13C: MOVEI S1,0 ;Load 0 for no message output
 MOVEI S2,[ASCIZ/Message Test (4 bytes of Image Data with 2 second delay)/]
 \$CALL TRNODE ;Store that message please

;build generate message ctp packet

MOVEI S1,4
 MOVEM S1,ITERCT
 MOVEI S1,3
 MOVEM S1,GENFUN ; genfunct =3/genfIMAGE
 \$CALL BLDCTP

;TRANSMIT REQUEST MESSAGE

\$CALL XMTREQ ;Transmit request
 JRST TC13A ;OK
 ERROR1 (,,, <Timed out waiting for event pending after transmitting generate message
 >,,NNN)
 JRST TST13X
 ERROR1 (,,, <No send complete event after transmitting generate message>,,NNN)
 JRST TST13X
 ERROR1 (,,, <Timed out waiting for generate message response available flag>,,NNN)
 JRST TST13X

1970
1971
1972
1973 001767' 402 00 0 00 011430'
1974 001770' 402 00 0 00 011423'
1975 001771' 476 00 0 00 011422'
1976 001772' 260 17 0 00 011043'
1977
1978 001773' 254 00 0 00 002004'
1979 001774' 260 17 0 00 012203'
1980 001775' 254 00 0 00 002030'
1981 001776' 260 17 0 00 012223'
1982 001777' 254 00 0 00 002030'
1983 002000' 260 17 0 00 012243'
1984 002001' 254 00 0 00 002030'
1985
1986 002002' 260 17 0 00 013241'
1987 002003' 254 00 0 00 002030'
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997 002004' 200 01 0 00 001736*
1998 002005' 200 02 0 00 012314'
1999 002006' 201 03 0 00 000000
2000 002007' 201 05 0 00 011433'
2001 002010' 275 05 0 00 000001
2002 002011' 200 06 0 00 012504'
2003 002012' 201 07 0 00 000000

;RECEIVE RESPONSE MESSAGE

TC13A: SETZM EXPSTA
SETZM EXPACT
SETOM CNTFLG
\$CALL RCVRM

JRST TC13B
ERROR1 (-1,EXPRSP,RCVRSP,Unexpected generate message response opcode,,NNN)
JRST TST13W
ERROR1 (-1,EXPSTA,RCVSTA,Unexpected generate message response status,,NNN)
JRST TST13W
ERROR1 (-1,EXPACT,RCVACT,Unexpected generate message response count,,NNN)
JRST TST13W
ERROR1 (1,NUMREF,RNVNUM,Unexpected generate message response reference number,,NNN)

JRST TST13W

;DATA COMPARISON ROUTINE FOR IMAGDT.

; REGISTER USAGE FOR BUFFER DATA:
; P2=DATA HOLDING REG OF BUFFER S1=ADDRESS OF BUFFER
; S2=BYTE POINTER FOR BUFFER T1=BYTE COUNTER OF BUFFER
; REGISTER USAGE FOR IMAGE DATA:
; T2=DATA HOLDING REG OF IMAGE DATA T3=ADDRESS OF IMAGDT
; T4=BYTE POINTER FOR IMAGE DATA P1=BYTE COUNTER OF IMAGDT

TC13B: MOVE S1,RETBUF
MOVE S2,[POINT 8,3(S1),15] ; 1ST DATA BYTE TO BE EXAMINED-1.
MOVEI T1,0 ; RESET BUFFER BYTE COUNTER
MOVEI T3,IMAGDT
SUBI T3,1
MOVE T4,[POINT 8,(T3),31] ; 1ST IMAGE BYTE TO BE EXAMINED-1.
MOVEI P1,0 ; RESET IMAGE BYTE COUNTER.

2004
 2005
 2006
 2007 002013' 134 04 0 00 000006
 2008 002014' 271 07 0 00 000001
 2009 002015' 312 07 0 00 011432'
 2010 002016' 254 00 0 00 002021'
 2011 002017' 201 07 0 00 000000
 2012 002020' 200 06 0 00 012504'
 2013
 2014
 2015
 2016 002021' 134 10 0 00 000002
 2017
 2018
 2019
 2020 002022' 302 10 0 04 000000
 2021 002023' 260 17 0 00 012505'
 2022
 2023
 2024
 2025 002024' 271 03 0 00 000001
 2026 002025' 311 03 0 00 011510'
 2027 002026' 254 00 0 00 002030'
 2028 002027' 254 00 0 00 002013'
 2029
 2030
 2031
 2032 002030' 260 17 0 00 001746*
 2033
 2034
 2035
 2036 002031' 260 17 0 00 001520*
 2037
 2038
 2039
 2040 002032' 260 17 0 00 011306'
 2041 002033' 254 00 0 00 002036'
 2042 002034' 260 17 0 00 011650'
 2043 002035' 254 00 0 00 002040'
 2044
 2045
 2046
 2047 002036' 201 01 0 00 000764
 2048 002037' 260 17 0 00 001526*
 2049 002040' 263 17 0 00 000000

;GET IMAGE DATA BYTE.

TC13C: ILDB T2,T4 ; T2 = IMAGE DATA BYTE.
 ADDI P1,1 ; INCR IMAGE DATA COUNT.
 CAME P1,IMGLN ; TEST IF IMAGE LENGTH DONE.
 JRST TC13D ; NO, CONTINUE WITH CURRENT COMPARE.
 MOVEI P1,0 ; RESET CURRENT IMAGE COUNT.
 MOVE T4,[POINT 8,(T3),31] ; RESET IMAGE DATA POINTER.

;GET BUFFER DATA BYTE.

TC13D: ILDB P2,S2 ; P2=BUFFER DATA BYTE

;COMPARE BUFFER DATA WITH IMAGE DATA.

CAIE P2,0(T2)
 ERROR1 (1,T2,P2,Data compare error,,NNNBYT)

;COMPARE BUFFER BYTE COUNT WITH COMMAND BYTE COUNT.

ADDI T1,1 ; INCR PRESENT BYTE POINTER
 CAML T1,ITERCT ; TEST IF PRESENT = ISSUED BYTES.
 JRST TST13W ; JUMP IF ALL BUFFER TESTED.
 JRST TC13C ; CONTINUE 'TIL ALL DATA BLOCK TESTED.

;Requeue the buffer

TST13W: \$CALL REQUEM

;Disconnect with selected node

TST13X: \$CALL DODIS ;Disconnect with the selected node

;Wait for connection closed

\$CALL CONCLD
 JRST TST13Y
 ERROR1 (,,Connection closed not in disconnect status,,NNN)
 JRST TST13Z

;Wait

TST13Y: MOVEI S1,^D500
 \$CALL WAITX
 TST13Z: \$RET

SUBTTL TST14 - Message Test - 1 to max bytes each of 4 data types

 * TST14 - MESSAGE TEST. THIS TEST WILL SEND A GROUP OF MESSAGE REQUESTS OF
 DATA=373, LENGTH=VARIABLE 1 TO MAX. , DELAY=0, REPEAT COUNT=0.

TST14: MOVEI S1,[ASCIZ/TST14/] ;Load test name
 MOVEI S2,[ASCIZ/Message Test (Variable length to max, data=373)/]
 \$CALL TRNODE ;Print node number if trace switch set

;Set starting count=0

SETZM ITERCT

;Connect with selected node.

\$CALL DOCONX

;Connect with the selected node

;Decide max message length.

MOVE S1,RSPMXM
 JUMPE S1,T14A
 JUMPL S1,T14A
 CAMLE S1,MINMS
 JRST T14A
 MOVE S1,RSPMXM
 MOVEM S1,MAXMSG
 JRST T14B

; MAX RESPONDER MSG LENGTH
 ; RSPNDR MS LENGTH=0, USE OURS.
 ; RSPNDR MS LENGTH NEG, USE OURS.
 ; TEST RSPNDR LFSSTHAN/EQUAL TO OURS
 ; JMP IF RSPNDR GREATER THAN OURS.
 ; USE RESPNDR'S MAX MSG LENGTH

T14A: MOVE S1,MINMS ; USE OUR MAX MSG LENGTH
 MOVEM S1,MAXMSG

;TEST FOR GOOD CONNECTION

T14B: \$CALL TEVPND ; TEST FOR EVENT PENDING.
 JRST T14C ; NO FAILURE,CONTINUE
 ERROR1 (,,,Timed out waiting for event pending flag during connection attempt,,NNN)

JRST TST14Y

T14C: \$CALL CONACC ; TEST FOR CONN ACCEPTED.
 JRST T14D ; NO FAILURE,CONTINUE
 ERROR1 (,,,Event pending was not connection accepted during connection attempt,,NNN)

JRST TST14Y

T14D: \$CALL CONOPN ; TEST FOR CONNECTION OPEN
 JRST T14E ; NO FAILURE,CONTINUE
 ERROR1 (,,,Connection not open after connection accepted,,NNN)
 JRST TST14Y

2050
 2051
 2052
 2053
 2054
 2055
 2056
 2057 002041' 201 01 0 00 013250'
 2058 002042' 201 02 0 00 013252'
 2059 002043' 260 17 0 00 011372'
 2060
 2061
 2062
 2063 002044' 402 00 0 00 011510'
 2064
 2065
 2066
 2067 002045' 260 17 0 00 000000*
 2068
 2069
 2070
 2071 002046' 200 01 0 00 000000*
 2072 002047' 322 01 0 00 002056'
 2073 002050' 321 01 0 00 002056'
 2074 002051' 313 01 0 00 000000*
 2075 002052' 254 00 0 00 002056'
 2076 002053' 200 01 0 00 002046*
 2077 002054' 202 01 0 00 011507'
 2078 002055' 254 00 0 00 002060'
 2079
 2080 002056' 201 01 0 00 002051*
 2081 002057' 202 01 0 00 011507'
 2082
 2083
 2084
 2085 002060' 260 17 0 00 011242'
 2086 002061' 254 00 0 00 002064'
 2087
 2088 002062' 260 17 0 00 011562'
 2089 002063' 254 00 0 00 002462'
 2090
 2091 002064' 260 17 0 00 011253'
 2092 002065' 254 00 0 00 002070'
 2093
 2094 002066' 260 17 0 00 011607'
 2095 002067' 254 00 0 00 002462'
 2096
 2097 002070' 260 17 0 00 011264'
 2098 002071' 254 00 0 00 002074'
 2099 002072' 260 17 0 00 011630'
 2100 002073' 254 00 0 00 002462'

2101
 2102
 2103
 2104 002074' 350 01 0 00 011510'
 2105 002075' 317 01 0 00 011507'
 2106 002076' 254 00 0 00 002101'
 2107 002077' 260 17 0 00 013272'
 2108 002100' 254 00 0 00 002455'
 2109
 2110
 2111
 2112 002101' 201 01 0 00 000004
 2113 002102' 202 01 0 00 001544*
 2114 002103' 200 01 0 00 011510'
 2115 002104' 202 01 0 00 001545*
 2116 002105' 201 01 0 00 000000
 2117 002106' 202 01 0 00 001622*
 2118 002107' 202 01 0 00 001551*
 2119 002110' 202 01 0 00 001755*
 2120 002111' 201 01 0 00 000373
 2121 002112' 202 01 0 00 001624*
 2122 002113' 260 17 0 00 001756*
 2123
 2124
 2125
 2126 002114' 260 17 0 00 011111'
 2127 002115' 254 00 0 00 002124'
 2128
 2129 002116' 260 17 0 00 013301'
 2130 002117' 254 00 0 00 002455'
 2131 002120' 260 17 0 00 013310'
 2132 002121' 254 00 0 00 002455'
 2133
 2134 002122' 260 17 0 00 013317'
 2135 002123' 254 00 0 00 002455'
 2136
 2137
 2138
 2139 002124' 402 00 0 00 011430'
 2140 002125' 402 00 0 00 011423'
 2141 002126' 476 00 0 00 011422'
 2142 002127' 260 17 0 00 011043'
 2143 002130' 254 00 0 00 002141'
 2144 002131' 260 17 0 00 013326'
 2145 002132' 254 00 0 00 002156'
 2146 002133' 260 17 0 00 013335'
 2147 002134' 254 00 0 00 002156'
 2148 002135' 260 17 0 00 013344'
 2149 002136' 254 00 0 00 002156'
 2150
 2151 002137' 260 17 0 00 013353'
 2152 002140' 254 00 0 00 002156'

;Increment iteration count.

T14E: AOS S1,ITERCT ;Bump message size
 CAMG S1,MAXMSG
 JRST T14E1 ; MAX MSG SIZE EXCEEDED,EXIT TEST.
 ERROR1 (2,MAXMSG,ITERCT,Maximum message size exceeded,,NNN)
 JRST TST14X

;Build generate message ctp packet

T14E1: MOVEI S1,4
 MOVEM S1,OPCODE ; OP CODE = GEN MESSAGE.
 MOVE S1,ITERCT
 MOVEM S1,GENLEN ; GEN LENGTH=4.
 MOVEI S1,0
 MOVEM S1,DELAY ; DELAY = 0.
 MOVEM S1,RCOUNT ; REPEAT COUNT = 0.
 MOVEM S1,GENFUN ; GENFUNCT =0/GENFILL
 MOVEI S1,373
 MOVEM S1,GNCNST ; GENCONST=373.
 \$CALL BLDCTP

;TRANSMIT REQUEST MESSAGE

\$CALL XMTREQ ;Transmit request
 JRST T14F ;OK
 ERROR1 (,,,<Timed out waiting for event pending after transmitting generate message
 >,,NNNSIZ)
 JRST TST14X
 ERROR1 (,,,<No send complete event after transmitting generate message>,,NNNSIZ)
 JRST TST14X
 ERROR1 (,,,<Timed out waiting for generate message response available flag>,,NNNSIZ
)
 JRST TST14X

;RECEIVE RESPONSE MESSAGE

T14F: SETZM EXPSTA
 SETZM EXPACT
 SETOM CNTFLG
 \$CALL RCVRM
 JRST T14G
 ERROR1 (-1,EXPRSP,RCVRSP,Unexpected generate message response opcode,,NNNSIZ)
 JRST T14I
 ERROR1 (-1,EXPSTA,RCVSTA,Unexpected generate message response status,,NNNSIZ)
 JRST T14I
 ERROR1 (-1,EXPACT,RCVACT,Unexpected generate message response count,,NNNSIZ)
 JRST T14I
 ERROR1 (1,NUMREF,RNVNUM,Unexpected generate message response reference number,,NNNS
 IZ)
 JRST T14I

2153
 2154
 2155
 2156
 2157 002141' 402 00 0 00 000003
 2158 002142' 200 01 0 00 002004*
 2159 002143' 200 02 0 00 013362'
 2160 002144' 135 06 0 00 000002
 2161 002145' 312 06 0 00 002112*
 2162 002146' 260 17 0 00 013363'
 2163
 2164
 2165
 2166 002147' 271 03 0 00 000001
 2167 002150' 311 03 0 00 011510'
 2168 002151' 254 00 0 00 002156'
 2169 002152' 134 06 0 00 000002
 2170 002153' 312 06 0 00 002145*
 2171 002154' 260 17 0 00 013372'
 2172 002155' 254 00 0 00 002147'
 2173
 2174
 2175
 2176 002156' 260 17 0 00 002030*
 2177
 2178
 2179
 2180 002157' 200 01 0 00 011510'
 2181 002160' 312 01 0 00 011507'
 2182 002161' 254 00 0 00 002163'
 2183 002162' 254 00 0 00 002166'
 2184
 2185
 2186
 2187 002163' 201 01 0 00 000144
 2188 002164' 260 17 0 00 002037*
 2189 002165' 254 00 0 00 002074'

```
;COMPARE DATBUFILL
;S1=DATA HOLDING REG,S2=DATA POINTER REG,T1=BYTE COUNTER

T14G: SETZM T1 ; T1=BYTE COUNTER OF DATA READ BACK.
      MOVE S1,RETBUF
      MOVE S2,[POINT 8,3(S1),23] ; 1ST DATA BYTE TO BE EXAMINED.
      LDB T4,S2 ; 1ST DATA BYTE TO BE EXAMINED.
      CAME T4,GNCNST ; TST IF COMPARE OK.
      ERROR1 (1,GNCNST,T4,Data compare error,,NNNBYL)

;TEST IF BYTE COUNT DATA READ BACK = TOTAL BYTES.

T14H: ADDI T1,1 ; INCR PRESENT BYTE POINTER
      CAML T1,ITERCT ; TEST IF PRESENT = ISSUED BYTES.
      JRST T14I ; JUMP IF ALL BUFFER TESTED.
      ILDB T4,S2 ; ELSE, NCONTINUE TO TEST DATA.
      CAME T4,GNCNST ; COMPARE DATA RECEIVED=EXPECTED.
      ERROR1 (1,GNCNST,T4,Data compare error,,NNNBYL)
      JRST T14H ; CONTINUE 'TIL ALL DATA BLOCK TESTED.

;Requeue the buffer

T14I: $CALL REQUEM

;LOOPBACK TO DO ALL ITERATIONS.

      MOVE S1,ITERCT
      CAME S1,MAXMSG ; Test present byte cnt = max byte cnt.
      JRST T14O ; NO CONTINUE IN LOOP.
      JRST TST14A ; YES, CONTINUE to next subTEST.

;WAIT .1 SECONDS BEFORE CONTINUING.

T14O: MOVEI S1,^D100
      $CALL WAITX
      JRST T14E
```

2190
 2191
 2192
 2193
 2194
 2195
 2196
 2197 002166' 201 01 0 00 000000
 2198 002167' 201 02 0 00 013401'
 2199 002170' 260 17 0 00 011372'
 2200
 2201
 2202
 2203 002171' 402 00 0 00 011510'
 2204
 2205
 2206
 2207 002172' 350 00 0 00 011510'
 2208 002173' 350 01 0 00 011510'
 2209 002174' 317 01 0 00 011507'
 2210 002175' 254 00 0 00 002200'
 2211 002176' 260 17 0 00 013272'
 2212 002177' 254 00 0 00 002455'
 2213
 2214
 2215
 2216 002200' 200 01 0 00 011510'
 2217 002201' 202 01 0 00 002104*
 2218
 2219 002202' 201 01 0 00 000000
 2220 002203' 202 01 0 00 002153*
 2221
 2222 002204' 201 01 0 00 000001
 2223 002205' 202 01 0 00 002110*
 2224
 2225 002206' 260 17 0 00 002113*
 2226
 2227
 2228
 2229
 2230 002207' 260 17 0 00 011111'
 2231 002210' 254 00 0 00 002217'
 2232
 2233 002211' 260 17 0 00 013301'
 2234 002212' 254 00 0 00 002455'
 2235 002213' 260 17 0 00 013310'
 2236 002214' 254 00 0 00 002455'
 2237
 2238 002215' 260 17 0 00 013317'
 2239 002216' 254 00 0 00 002455'

```

*****
* TST14A - MESSAGE TEST. THIS TEST WILL SEND A GROUP OF MESSAGE REQUESTS OF
* DATA=BYTEPAIR, LENGTH=VARIABLE 1 TO MAX., DELAY=0, REPEAT COUNT=0.
* ###N.B.-THIS CODE MUST HAVE EVEN BYTE COUNTS.###
*****

TST14A: MOVEI S1,0 ;Load 0 for no message output
        MOVEI S2,[ASCIZ/Message Test (Variable length to max, data=byte pairs)/]
        $CALL TRNODE ;Store that message please

;SET STARTING COUNT =0

        SETZM ITERCT

;Increment iteration cnt.

TA14A: AOS ITERCT
        AOS S1,ITERCT
        CAMG S1,MAXMSG
        JRST TA14A1 ; IF HERE,MAX MSG SIZE EXCEEDED
        ERROR1 (2,MAXMSG,ITERCT,Maximum message size exceeded,,NNN)
        JRST TST14X ; EXIT TEST.

;build generate message ctp packet

TA14A1: MOVE S1,ITERCT
        MOVEM S1,GENLEN ; GEN LENGTH=ITERCT

        MOVEI S1,0
        MOVEM S1,GENCONST ; GENCONST=377.

        MOVEI S1,1
        MOVEM S1,GENFUNCT ; GENFUNCT =0/GENFBPAIR

        $CALL BLDCTP

;TRANSMIT REQUEST MESSAGE

        $CALL XMTREQ ;Transmit request
        JRST TA14B ;OK
        ERROR1 (,,<Timed out waiting for event pending after transmitting generate message
>,,NNNSIZ)
        JRST TST14X
        ERROR1 (,,<No send complete event after transmitting generate message>,,NNNSIZ)
        JRST TST14X
        ERROR1 (,,<Timed out waiting for generate message response available flag>,,NNNSIZ)
        JRST TST14X
  
```



```

2240
2241
2242
2243 002217' 402 00 0 00 011430'
2244 002220' 402 00 0 00 011423'
2245 002221' 476 00 0 00 011422'
2246 002222' 260 17 0 00 011043'
2247
2248 002223' 254 00 0 00 002234'
2249 002224' 260 17 0 00 013326'
2250 002225' 254 00 0 00 002261'
2251 002226' 260 17 0 00 013335'
2252 002227' 254 00 0 00 002261'
2253 002230' 260 17 0 00 013344'
2254 002231' 254 00 0 00 002261'
2255
2256 002232' 260 17 0 00 013414'
2257 002233' 254 00 0 00 002261'
2258
2259
2260
2261
2262 002234' 402 00 0 00 000003'
2263 002235' 200 01 0 00 002142'
2264 002236' 200 06 0 00 012314'
2265 002237' 403 10 0 00 000011'
2266 002240' 254 00 0 00 002244'
2267
2268
2269
2270 002241' 271 11 0 00 000001'
2271 002242' 622 11 0 00 000400'
2272 002243' 271 10 0 00 000001'
2273
2274
2275
2276 002244' 271 03 0 00 000001'
2277 002245' 134 04 0 00 000006'
2278 002246' 134 02 0 00 000006'
2279 002247' 302 11 0 04 000000'
2280 002250' 260 17 0 00 013423'
2281 002251' 316 03 0 00 011510'
2282 002252' 254 00 0 00 002261'
2283 002253' 271 03 0 00 000001'
2284 002254' 302 10 0 02 000000'
2285 002255' 260 17 0 00 013432'
2286 002256' 316 03 0 00 011510'
2287 002257' 254 00 0 00 002261'
2288 002260' 254 00 0 00 002241'

```

;RECEIVE RESPONSE MESSAGE

```

TA14B: SETZM EXPSTA
        SETZM EXPACT
        SETOM CNTFLG
        $CALL RCVRM

        JRST TA14C
ERROR1 (-1,EXPRSP,RCVRSP,Unexpected generate message response opcode,,NNNSIZ)
        JRST TA14F
ERROR1 (-1,EXPSTA,RCVSTA,Unexpected generate message response status,,NNNSIZ)
        JRST TA14F
ERROR1 (-1,EXPACT,RCVACT,Unexpected generate message response count,,NNNSIZ)
        JRST TA14F
ERROR1 (1,NUMREF,RNVNUM,Unexpected generate message response reference number,,NNNS
12)
        JRST TA14F

```

```

;COMPARE DATA=BYTEPAIR
;S1=BUFFER ADDRESS HOLDING REG,T4=DATA POINTER REG,T1=BYTE COUNTER

```

```

TA14C: SETZM T1 ; T1=BYTE COUNTER WITHIN BUFFER.
        MOVE S1,RETBUF
        MOVE T4,[POINT 8,3(S1),15] ; POINTER TO DATA BYTE TO BE EXAMINED.
        CLEARB P2,P3 ; CLEAR COMPARAND REGS.
        JRST TA14E ; SKIP OVER 1ST INCR(KEEP ZERO)

```

;INCREMENT COMPARAND REGS

```

TA14D: ADDI P3,1 ; INCREMENT L/S BYTE OF BYTEPAIR.
        TRZE P3,400 ; TEST IF INCREMENT M/S BYTE NECESSARY.
        ADDI P2,1 ; INCREMENT M/S BYTE.

```

;COMPARE DATA BYTES

```

TA14E: ADDI T1,1 ; INCR PRESENT BYTE CNTR.
        ILDB T2,T4 ; LOAD S1 = L/S BYTE COMPARAND(BUFFER)
        ILDB S2,T4 ; LOAD S2 = M/S BYTE COMPARAND(BUFFER)
        CAIE P3,0(T2) ; COMPARE LS BYTE.
ERROR1 (1,P3,T2,Data compare error on least significant byte,,NNNBYL)
        CAMN T1,ITERCT ; TEST IF ALL BUFFER TESTED.
        JRST TA14F ; BUFFER TESTED,EXIT TEST SEGMENT.
        ADDI T1,1 ; CONTINUE TESTING BUFFER/INCR BYTE(CNT.
        CAIE P2,0(S2) ; COMPARE MS BYTE.
ERROR1 (1,P2,S2,Data compare error on most significant byte,,NNNBYL)
        CAMN T1,ITERCT ; TEST IF ALL BUFFER TESTED.
        JRST TA14F ; END OF BUFFER JUMP HERE.
        JRST TA14D ; CONTINUE IN BUFFER COMPARE LOOP.

```


77

DFCIBM CTP Tests 01-80 version 2(20)
DFCIBM MAC 22-Aug-85 20:30MACRO %53B(1242) 17:38 27-Aug-85 Page 51
TST14 - Message Test - 1 to max bytes each of 4 data types

SEQ 0390

2289
2290
2291
2292 002261' 260 17 0 00 002156*
2293
2294
2295
2296 002262' 312 03 0 00 011507'
2297 002263' 254 00 0 00 002265'
2298 002264' 254 00 0 00 002270'
2299
2300
2301
2302
2303 002265' 201 01 0 00 000144
2304 002266' 260 17 0 00 002164*
2305 002267' 254 00 0 00 002172'

;Requeue the buffer

TA14F: \$CALL REQUEM

;LOOPBACK TO DO ALL ITERATIONS.

CAME T1,MAXMSG
JRST TA14G
JRST TST14B

; TEST IF PAST BYTE CNT = MAX BYTE CNT.
; NO CONTINUE IN LOOP.
; BUFFER TESTED,EXIT TEST SEGMENT.

;WAIT .1 SECONDS BEFOR CONTINUING.

TA14G: MOVEI S1,^D100
\$CALL WAITX
JRST TA14A

NNN)

NNN

age

N)

2306
 2307
 2308
 2309
 2310
 2311
 2312
 2313 002270' 201 01 0 00 000000
 2314 002271' 201 02 0 00 013441'
 2315 002272' 260 17 0 00 011372'
 2316
 2317
 2318
 2319 002273' 402 00 0 00 011510'
 2320
 2321
 2322
 2323 002274' 350 01 0 00 011510'
 2324 002275' 317 01 0 00 011507'
 2325 002276' 254 00 0 00 002301'
 2326 002277' 260 17 0 00 013272'
 2327 002300' 254 00 0 00 002455'
 2328
 2329
 2330
 2331 002301' 200 01 0 00 011510'
 2332 002302' 202 01 0 00 002201*
 2333
 2334 002303' 201 01 0 00 000002
 2335 002304' 202 01 0 00 002205*
 2336
 2337 002305' 260 17 0 00 002206*
 2338
 2339
 2340
 2341 002306' 260 17 0 00 011111'
 2342 002307' 254 00 0 00 002316'
 2343
 2344 002310' 260 17 0 00 013301'
 2345 002311' 254 00 0 00 002455'
 2346 002312' 260 17 0 00 013310'
 2347 002313' 254 00 0 00 002455'
 2348
 2349 002314' 260 17 0 00 013317'
 2350 002315' 254 00 0 00 002455'

```

*****
* TST14L - MESSAGE TEST. THIS TEST WILL SEND A GROUP OF MESSAGE REQUESTS
* OF DATA=RESPONDER NODE,LENGTH=VARIABLE 1 TO MAX.,DELAY=0,
* REPEAT COUNT=0.
*****

TST14B: MOVEI    S1,0                ;Load 0 for no message output
        MOVEI    S2,[ASCIZ/Message Test (Variable length to max, Data=responder node)/]
        $CALL    TRNODE              ;Store that message please

;SET STARTING COUNT =0

        SETZM    ITERCT

;Increment iteration cnt.

TB14A:  AOS      S1,ITERCT            ;Bump count
        CAMG     S1,MAXMSG
        JRST     TB14A1              ; IF HERE,MAX MSG SIZE EXCEEDED
        ERROR1    (2,MAXMSG,ITERCT,Maximum message size exceeded,,NNN)
        JRST     TST14X              ; EXIT TEST.

;Build generate message CTP packet

TB14A1: MOVE     S1,ITERCT
        MOVEM    S1,GENLEN            ; GEN LENGTH=4.

        MOVEI    S1,2
        MOVEM    S1,GENFUN            ; GENFUNCT =0/GENFNODE

        $CALL    BLDCTP

;TRANSMIT REQUEST MESSAGE

        $CALL    XMTREQ                ;Transmit request
        JRST     TB14B                ;OK
        ERROR1    (,,,<Timed out waiting for event pending after transmitting generate message
>,,NNNSIZ)
        JRST     TST14X
        ERROR1    (,,,<No send complete event after transmitting generate message>,,NNNSIZ)
        JRST     TST14X
        ERROR1    (,,,<Timed out waiting for generate message response available flag>,,NNNSIZ
)
        JRST     TST14X

```

2351
 2352
 2353
 2354 002316' 402 00 0 00 011430'
 2355 002317' 402 00 0 00 011423'
 2356 002320' 476 00 0 00 011422'
 2357 002321' 260 17 0 00 011043'
 2358
 2359 002322' 254 00 0 00 002333'
 2360 002323' 260 17 0 00 013326'
 2361 002324' 254 00 0 00 002344'
 2362 002325' 260 17 0 00 013335'
 2363 002326' 254 00 0 00 002344'
 2364 002327' 260 17 0 00 013344'
 2365 002330' 254 00 0 00 002344'
 2366
 2367 002331' 260 17 0 00 013455'
 2368 002332' 254 00 0 00 002344'
 2369
 2370
 2371
 2372
 2373 002333' 402 00 0 00 000003
 2374 002334' 200 01 0 00 002235*
 2375 002335' 200 02 0 00 012314'
 2376
 2377 002336' 271 03 0 00 000001
 2378 002337' 134 06 0 00 000002
 2379 002340' 312 06 0 00 001741*
 2380 002341' 260 17 0 00 013464'
 2381 002342' 312 03 0 00 011510'
 2382 002343' 254 00 0 00 002336'
 2383
 2384
 2385
 2386 002344' 260 17 0 00 002261*
 2387
 2388
 2389
 2390 002345' 312 03 0 00 011507'
 2391 002346' 254 00 0 00 002350'
 2392 002347' 254 00 0 00 002353'
 2393
 2394
 2395
 2396 002350' 201 01 0 00 000144
 2397 002351' 260 17 0 00 002266*
 2398 002352' 254 00 0 00 002274'

;RECEIVE RESPONSE MESSAGE

TB14B: SETZM EXPSTA
 SETZM EXPACT
 SETOM CNTFLG
 \$CALL RCVRM
 JRST TB14C
 ERROR1 (-1,EXPRSP,RCVRSP,Unexpected generate message response opcode,,NNNSIZ)
 JRST TB14E
 ERROR1 (-1,EXPSTA,RCVSTA,Unexpected generate message response status,,NNNSIZ)
 JRST TB14E
 ERROR1 (-1,EXPACT,RCVACT,Unexpected generate message response count,,NNNSIZ)
 JRST TB14E
 ERROR1 (1,NUMREF,RNVNUM,Unexpected generate message response reference number,,NNNS
 IZ)
 JRST TB14E

;COMPARE DATBUFILL
 ;S1=DATA HOLDING REG,S2=DATA POINTER REG,T1=BYTE COUNTER

TB14C: SETZM T1 ; T1=BYTE COUNTER OF DATA READ BACK.
 MOVE S1,RETBUF
 MOVE S2,[POINT 8,3(S1),15] ; 1ST DATA BYTE TO BE EXAMINED.
 TB14D: ADDI T1,1 ; INCR BYTE COUNT.
 ILDB T4,S2 ; LOAD DATA BYTE FROM BUFFER
 CAME T4,NODEN ; COMPARE TO NODE NUMBER.
 ERROR1 (1,NODEN,T4,Data compare error,,NNNBYL)
 CAME T1,ITERCT ; TEST IF ALL BUFFER TESTED.
 JRST TB14D ; CONTINUE IN DATA COMPARE LOOP.

;Requeue the buffer

TB14E: \$CALL REQUEM

;LOOPBACK TO DO ALL ITERATIONS.

CAME T1,MAXMSG ; TEST IF PAST BYTE CNT = MAX BYTE CNT.
 JRST TB14F ; NO CONTINUE IN LOOP.
 JRST TST14C ; YES, CONTINUE IN TEST.

;WAIT .1 SECONDS BEFOR CONTINUING.

TB14F: MOVEI S1,^D100
 \$CALL WAITX
 JRST TB14A

2399
 2400
 2401
 2402
 2403
 2404
 2405 002353' 201 01 0 00 000000
 2406 002354' 201 02 0 00 013473'
 2407 002355' 260 17 0 00 011372'
 2408
 2409
 2410
 2411 002356' 402 00 0 00 011510'
 2412
 2413 002357' 350 01 0 00 011510'
 2414 002360' 317 01 0 00 011507'
 2415 002361' 254 00 0 00 002364'
 2416 002362' 260 17 0 00 013272'
 2417 002363' 254 00 0 00 002455'
 2418
 2419
 2420
 2421 002364' 200 01 0 00 011510'
 2422 002365' 202 01 0 00 002302*
 2423
 2424 002366' 200 01 0 00 012474'
 2425 002367' 202 01 0 00 011433'
 2426 002370' 201 01 0 00 000004
 2427 002371' 202 01 0 00 011432'
 2428
 2429 002372' 201 01 0 00 000003
 2430 002373' 202 01 0 00 002304*
 2431
 2432 002374' 260 17 0 00 002305*
 2433
 2434
 2435
 2436 002375' 260 17 0 00 011111'
 2437 002376' 254 00 0 00 002405'
 2438
 2439 002377' 260 17 0 00 013301'
 2440 002400' 254 00 0 00 002455'
 2441 002401' 260 17 0 00 013310'
 2442 002402' 254 00 0 00 002455'
 2443
 2444 002403' 260 17 0 00 013317'
 2445 002404' 254 00 0 00 002455'

 * TST14C - MESSAGE TEST. THIS TEST WILL SEND A GROUP OF MESSAGE REQUESTS OF
 DATA=IMAGEDATA, LENGTH=VARIABLE 1 TO MAX. , DELAY=0, REPEAT COUNT=0.

TST14C: MOVEI S1,0 ;Load 0 for no message output
 MOVEI S2,[ASCIZ/Message Test (Variable length to max, Data=Image Data)/]
 \$CALL TRNODE ;Store that message please

;SET STARTING COUNT =0

SETZM ITERCT
 TC14A: AOS S1,ITERCT ;Bump ITERCT and move it to S1
 CAMG S1,MAXMSG ;Max message size exceeded?
 JRST TC14A1 ;Nope
 ERROR1 (2,MAXMSG,ITERCT,Maximum message size exceeded,,NNN)
 JRST TST14X ; EXIT TEST.

;Build generate message CTP packet

TC14A1: MOVE S1,ITERCT
 MOVEM S1,GENLEN
 MOVE S1,[777773757700] ; IMAGE=377,376,375,374.
 MOVEM S1,IMAGDT
 MOVEI S1,4
 MOVEM S1,IMGLN ; IMAGE LENGTH=4.
 MOVEI S1,3
 MOVEM S1,GENFUN ; GENFUNCT =3/GENFIMAGE

\$CALL BLDCTP

;TRANSMIT REQUEST MESSAGE

\$CALL XMTREQ ;Transmit request
 JRST TC14B ;OK
 ERROR1 (,,, <Timed out waiting for event pending after transmitting generate message
 >,,NNNSIZ)
 JRST TST14X
 ERROR1 (,,, <No send complete event after transmitting generate message>,,NNNSIZ)
 JRST TST14X
 ERROR1 (,,, <Timed out waiting for generate message response available flag>,,NNNSIZ
)
 JRST TST14X

2446
 2447
 2448
 2449 002405' 402 00 0 00 011430'
 2450 002406' 402 00 0 00 011423'
 2451 002407' 476 00 0 00 011422'
 2452 002410' 260 17 0 00 011043'
 2453
 2454 002411' 254 00 0 00 002422'
 2455 002412' 260 17 0 00 013326'
 2456 002413' 254 00 0 00 002446'
 2457 002414' 260 17 0 00 013335'
 2458 002415' 254 00 0 00 002446'
 2459 002416' 260 17 0 00 013344'
 2460 002417' 254 00 0 00 002446'
 2461
 2462 002420' 260 17 0 00 013506'
 2463 002421' 254 00 0 00 002446'
 2464
 2465
 2466
 2467
 2468
 2469
 2470
 2471
 2472
 2473 002422' 200 01 0 00 002334*
 2474 002423' 200 02 0 00 012314'
 2475 002424' 201 03 0 00 000000
 2476 002425' 201 05 0 00 011433'
 2477 002426' 275 05 0 00 000001
 2478 002427' 200 06 0 00 012504'
 2479 002430' 201 07 0 00 000000
 2480
 2481
 2482
 2483 002431' 134 04 0 00 000006
 2484 002432' 271 07 0 00 000001
 2485 002433' 312 07 0 00 011432'
 2486 002434' 254 00 0 00 002437'
 2487 002435' 201 07 0 00 000000
 2488 002436' 200 06 0 00 012504'

;RECEIVE RESPONSE MESSAGE

TC14B: SETZM EXPSTA
 SETZM EXPACT
 SETOM CNTFLG
 \$CALL RCVRM
 JRST TC14C
 ERROR1 (-1,EXPRSP,RCVRSP,Unexpected generate message response opcode,,NNNSIZ)
 JRST TC14F
 ERROR1 (-1,EXPSTA,RCVSTA,Unexpected generate message response status,,NNNSIZ)
 JRST TC14F
 ERROR1 (-1,EXPACT,RCVACT,Unexpected generate message response count,,NNNSIZ)
 JRST TC14F
 ERROR1 (1,NUMREF,RNVNUM,Unexpected generate message response reference number,,NNNS
 IZ)
 JRST TC14F

;DATA COMPARISON ROUTINE FOR IMAGDT.

; REGISTER USAGE FOR BUFFER DATA:
 ; P2=DATA HOLDING REG OF BUFFER S1=ADDRESS OF BUFFER
 ; S2=BYTE POINTER FOR BUFFER T1=BYTE COUNTER OF BUFFER
 ; REGISTER USAGE FOR IMAGE DATA:
 ; T2=DATA HOLDING REG OF IMAGE DATA T3=ADDRESS OF IMAGDT
 ; T4=BYTE POINTER FOR IMAGE DATA P1=BYTE COUNTER OF IMAGDT

TC14C: MOVE S1,RETBUF
 MOVE S2,[POINT 8,3(S1),15] ; 1ST DATA BYTE TO BE EXAMINED-1.
 MOVEI T1,0 ; RESET BUFFER BYTE COUNTER
 MOVEI T3,IMAGDT
 SUBI T3,1
 MOVE T4,[POINT 8,(T3),31] ; 1ST IMAGE BYTE TO BE EXAMINED-1.
 MOVEI P1,0 ; RESET IMAGE BYTE COUNTER.

;GET IMAGE DATA BYTE.

TC14D: ILDB T2,T4 ; T2 = IMAGE DATA BYTE.
 ADDI P1,1 ; INCR IMAGE DATA COUNT.
 CAME P1,IMGLN ; TEST IF IMAGE LENGTH DONE.
 JRST TC14E ; NO, CONTINUE WITH CURRENT COMPARE.
 MOVEI P1,0 ; RESET CURRENT IMAGE COUNT.
 MOVE T4,[POINT 8,(T3),31] ; RESET IMAGE DATA POINTER.


```

2489                                     ;GET BUFFER DATA BYTE.
2490
2491                                     ; P2=BUFFER DATA BYTE
2492 002437' 134 10 0 00 000002      TC14E: ILDB      P2,S2
2493                                     ;COMPARE BUFFER DATA WITH IMAGE DATA.
2494
2495                                     CAIE      P2,0(T2)
2496 002440' 302 10 0 04 000000      ERROR1 (1,T2,P2,Data compare error,,NNNBYL)
2497 002441' 260 17 0 00 013515'
2498
2499                                     ;COMPARE BUFFER BYTE COUNT WITH COMMAND BYTE COUNT.
2500
2501 002442' 271 03 0 00 000001      ADDI      T1,1          ; INCR PRESENT BYTE PCINTER
2502 002443' 311 03 0 00 011510'      CAML      T1,ITERCT      ; TEST IF PRESENT = ISSUED BYTES.
2503 002444' 254 00 0 00 002446'      JRST      TC14F          ; JUMP IF ALL BUFFER TESTED.
2504
2505 002445' 254 00 0 00 002431'      JRST      TC14D          ; CONTINUE 'TIL ALL DATA BLOCK TESTED.
2506
2507                                     ;Requeue the buffer
2508
2509 002446' 260 17 0 00 002344*      TC14F: $CALL  REQUEM
2510
2511                                     ;LOOPBACK TO DO ALL ITERATIONS.
2512
2513 002447' 312 03 0 00 011507'      CAME      T1,MAXMSG      ; TEST IF PAST BYTE CNT = MAX BYTE CNT.
2514 002450' 254 00 0 00 002452'      JRST      TC14G          ; NO CONTINUE IN LOOF.
2515 002451' 254 00 0 00 002455'      JRST      TST14X         ; YES, EXIT TEST.
2516
2517                                     ;WAIT .1 SECONDS BEFOR CONTINUING.
2518
2519 002452' 260 01 0 00 000144      TC14G: MOVEI    S1,^D100
2520 002453' 260 17 0 00 002351*      $CALL    WAITX
2521 002454' 254 00 0 00 002357'      JRST      TC14A
2522
2523                                     ;Disconnect with selected node
2524
2525 002455' 260 17 0 00 002031*      TST14X: $CALL  DODIS          ;Disconnect with the selected node
2526
2527                                     ;Wait for connection closed
2528
2529 002456' 260 17 0 00 011306'      $CALL    CONCLD
2530 002457' 254 00 0 00 002462'      JRST      TST14Y
2531 002460' 260 17 0 00 011650'      ERROR1 (,,Connection closed not in disconnect status,,NNN)
2532 002461' 254 00 0 00 002464'      JRST      TST14Z
2533
2534                                     ;Wait
2535
2536 002462' 201 01 0 00 000764      TST14Y: MOVEI    S1,^D500
2537 002463' 260 17 0 00 002453*      $CALL    WAITX
2538
2539 002464' 263 17 0 00 000000      TST14Z: $RET

```

SUBTTL TST30 - Basic Datagram Test

 * TST30 - BASIC DATAGRAM TEST. THIS TEST WILL SEND A DATAGRAM REQUEST OF
 DATA=0, LENGTH=0, DELAY=0, REPEAT COUNT=0.

TST30: MOVEI S1,[ASCIZ/TST30/] ;Load test name
 MOVEI S2,[ASCIZ/Basic Datagram Test/]
 \$CALL TRNODE ;Print node number if trace switch set

;Connect with selected node and verify connection=open.

\$CALL CONNCT
 JRST TST30A ; JUMP IF CONNECTION OK.
 ERROR1 (,,,Timed out waiting for event pending flag during connection attempt,,NNN)
 JRST TST30X
 ERROR1 (,,,Event pending was not connection accepted during connection attempt,,NNN)
)
 JRST TST30X
 ERROR1 (,,,Connection not open after connection accepted,,NNN)
 JRST TST30X

;Build generate DATAGRAM ctp packet

TST30A: MOVEI S1,5
 MOVEM S1,OPCODE ; opcode = gen DATAGRAM
 MOVEI S1,0
 MOVEM S1,GENFUN ; genfunct = 0/genfill.
 MOVEM S1,DELAY ; delay = 0.
 MOVEM S1,GENLEN ; gen length = 0.
 MOVEM S1,GNCNST ; genconst = 0.
 MOVEM S1,RCOUNT ; repeat count = 0.

\$CALL BLDCTP

;Send the DATAGRAM

\$CALL SNDDAT

;Wait for msg/dgm send complete and datagram available

\$CALL TEVPND ; Test for event pending.
 JRST TST30E ; No failure, continue
 ERROR1 (,,,Timed out waiting for event pending after generate datagram sent,,NNN)
 JRST TST30W

TST30E: \$CALL SNDCMP ; Test for msg/dgm send complete.
 JRST TST30F ; No failure, continue
 ERROR1 (,,,Time out waiting for generate datagram send complete,,NNN)
 JRST TST30W

2540
 2541
 2542
 2543
 2544
 2545
 2546
 2547 002465' 201 01 0 00 013524'
 2548 002466' 201 02 0 00 013526'
 2549 002467' 260 17 0 00 011372'
 2550
 2551
 2552
 2553 002470' 260 17 0 00 011134'
 2554 002471' 254 00 0 00 002500'
 2555
 2556 002472' 260 17 0 00 011562'
 2557 002473' 254 00 0 00 002556'
 2558
 2559 002474' 260 17 0 00 011607'
 2560 002475' 254 00 0 00 002556'
 2561 002476' 260 17 0 00 011630'
 2562 002477' 254 00 0 00 002556'
 2563
 2564
 2565
 2566 002500' 201 01 0 00 000005
 2567 002501' 202 01 0 00 002102*
 2568
 2569 002502' 201 01 0 00 000000
 2570 002503' 202 01 0 00 002373*
 2571 002504' 202 01 0 00 002106*
 2572 002505' 202 01 0 00 002365*
 2573 002506' 202 01 0 00 002203*
 2574 002507' 202 01 0 00 002107*
 2575
 2576 002510' 260 17 0 00 002374*
 2577
 2578
 2579
 2580 002511' 260 17 0 00 000000*
 2581
 2582
 2583
 2584 002512' 260 17 0 00 011242'
 2585 002513' 254 00 0 00 002516'
 2586 002514' 260 17 0 00 013547'
 2587 002515' 254 00 0 00 002552'
 2588
 2589 002516' 260 17 0 00 011317'
 2590 002517' 254 00 0 00 002522'
 2591 002520' 260 17 0 00 013571'
 2592 002521' 254 00 0 00 002552'

2593
 2594 002522' 260 17 0 00 011341'
 2595 002523' 254 00 0 00 002526'
 2596 002524' 260 17 0 00 013610'
 2597 002525' 254 00 0 00 002552'
 2598
 2599
 2600
 2601 002526' 260 17 0 00 000000*
 2602
 2603
 2604
 2605 002527' 201 03 0 00 000105
 2606 002530' 200 01 0 00 002422*
 2607 002531' 135 02 0 00 013617'
 2608 002532' 316 02 0 00 000003
 2609 002533' 254 00 0 00 002536'
 2610 002534' 260 17 0 00 013627'
 2611 002535' 254 00 0 00 002551'
 2612
 2613
 2614 002536' 201 03 0 00 000000
 2615 002537' 135 02 0 00 013636'
 2616 002540' 316 02 0 00 000003
 2617 002541' 254 00 0 00 002544'
 2618 002542' 260 17 0 00 013646'
 2619 002543' 254 00 0 00 002551'
 2620
 2621
 2622 002544' 201 03 0 00 000000
 2623 002545' 135 02 0 00 013655'
 2624 002546' 316 02 0 00 000003
 2625 002547' 254 00 0 00 002551'
 2626 002550' 260 17 0 00 013667'
 2627
 2628
 2629 002551' 260 17 0 00 000000*
 2630
 2631
 2632 002552' 260 17 0 00 002455*
 2633
 2634
 2635 002553' 260 17 0 00 011306'
 2636 002554' 254 00 0 00 002556'
 2637 002555' 260 17 0 00 011650'
 2638
 2639
 2640 002556' 201 01 0 00 000764
 2641 002557' 260 17 0 00 002463*
 2642 002560' 263 17 0 00 000000

TST30F: \$CALL DATAVL ; Wait for datagram available flag
 JRST TST30G ; No failure, continue
 ERROR1 (, , Time out waiting for generated datagram,,NNN)
 JRST TST30W

 ;Read the DATAGRAM
 TST30G: \$CALL READD

 ;Test response datagram opcode
 MOVEI T1,^D69 ;Make correct 69
 MOVE S1,RETBUF ;Get buffer address
 LDB S2,[POINT 8,(S1),7] ;Put returned opcode in S2
 CAMN S2,T1 ;Is returned status as expected ?
 JRST TST30H ; Yes
 ERROR1 (3,T1,S2,Response datagram opcode in error,,NNN)
 JRST TST30Y

 ;Test response datagram status
 TST30H: MOVEI T1,0 ;Set up correct
 LDB S2,[POINT 8,1(S1),15] ; S2=STATUS
 CAMN S2,T1 ;Is returned status as expected ?
 JRST TST30I ; Yes
 ERROR1 (3,T1,S2,Response datagram status in error,,NNN)
 JRST TST30Y

 ;Test act count = 0.
 TST30I: MOVEI T1,0 ;Set up correct
 LDB S2,[POINT 16,1(S1),31] ; s2= act cnt.
 CAMN S2,T1 ; test ACT CNT = EXPECTED.
 JRST TST30Y ; Yes
 ERROR1 (6,T1,S2,Response datagram account field in error,,NNN)

 ;Requeue the buffer
 TST30Y: \$CALL REQUED

 ;Disconnect with selected node
 TST30W: \$CALL DODIS ;Disconnect with the selected node

 ;Wait for connection closed
 \$CALL CONCLO
 JRST TST30X
 ERROR1 (, , ,Connection closed not in disconnect status,,NNN)

 ;Wait
 TST30X: MOVEI S1,^D500
 \$CALL WAITX
 \$RET

SUBTTL TST31 - Datagram 4 Bytes Each Of 4 Data Types

 ;* TST31 - DATAGRAM TEST. THIS TEST WILL SEND A DATAGRAM REQUEST OF
 ; DATA=377, LENGTH=4, DELAY=0, REPEAT COUNT=0.
 ;*****

TST31:

TS31A: MOVEI S1,[ASCIZ/TST31/] ;Load test name
 MOVEI S2,[ASCIZ/Test 31 Part A - Datagram Test (Data=377)/]
 \$CALL TRNODE ;Print node number if trace switch set

;Connect with selected node and verify connection=open.

\$CALL CONNCT
 JRST TS31A ; JUMP IF CONNECTION OK.
 ERROR1 (,,,Timed out waiting for event pending flag during connection attempt,,NNN)

JRST TST31X
 ERROR1 (,,,Event pending was not connection accepted during connection attempt,,NNN)

)

JRST TST31X
 ERROR1 (,,,Connection not open after connection accepted,,NNN)
 JRST TST31X

;Build generate datagram ctp packet

TST31A: MOVEI S1,5
 MOVEM S1,OPCODE ; OPCODE = GEN DATAGRAM
 MOVEM S1,GENLEN ; GEN LENGTH=4.

MOVEI S1,0
 MOVEM S1,DELAY ; DELAY = 0.
 MOVEM S1,RCOUNT ; REPEAT COUNT = 0.
 MOVEM S1,GENFUN ; GENFUNCT =0/GENFILL

MOVEI S1,377
 MOVEM S1,GNCNST ; GENCONST=377.

\$CALL BLDCTP

;Send the DATAGRAM

\$CALL SNDDAT

;Wait for msg/dgm send complete and datagram available

\$CALL TEVPND ; Test for event pending.
 JRST TS31AD ; No failure,continue
 ERROR1 (,,,Timed out waiting for event pending after generate datagram sent,,NNN)
 JRST TST31W

2643
 2644
 2645
 2646
 2647
 2648
 2649
 2650 002561' 201 01 0 00 013676'
 2651 002561' 201 01 0 00 013676'
 2652 002562' 201 02 0 00 013700'
 2653 002563' 260 17 0 00 011372'
 2654
 2655
 2656
 2657 002564' 260 17 0 00 011134'
 2658 002565' 254 00 0 00 002574'
 2659
 2660 002566' 260 17 0 00 011562'
 2661 002567' 254 00 0 00 003153'
 2662
 2663 002570' 260 17 0 00 011607'
 2664 002571' 254 00 0 00 003153'
 2665 002572' 260 17 0 00 011630'
 2666 002573' 254 00 0 00 003153'
 2667
 2668
 2669
 2670 002574' 201 01 0 00 000005
 2671 002575' 202 01 0 00 002501*
 2672 002576' 202 01 0 00 002505*
 2673
 2674 002577' 201 01 0 00 000000
 2675 002600' 202 01 0 00 002504*
 2676 002601' 202 01 0 00 002507*
 2677 002602' 202 01 0 00 002503*
 2678
 2679 002603' 201 01 0 00 000377
 2680 002604' 202 01 0 00 002506*
 2681
 2682 002605' 260 17 0 00 002510*
 2683
 2684
 2685
 2686 002606' 260 17 0 00 002511*
 2687
 2688
 2689
 2690 002607' 260 17 0 00 011242'
 2691 002610' 254 00 0 00 002613'
 2692 002611' 260 17 0 00 013547'
 2693 002612' 254 00 0 00 003147'

2694
 2695 002613' 260 17 0 00 011317'
 2696 002614' 254 00 0 00 002617'
 2697 002615' 260 17 0 00 013571'
 2698 002616' 254 00 0 00 003147'
 2699
 2700 002617' 260 17 0 00 011341'
 2701 002620' 254 00 0 00 002623'
 2702 002621' 260 17 0 00 013610'
 2703 002622' 254 00 0 00 003147'
 2704
 2705
 2706
 2707 002623' 260 17 0 00 002526*
 2708
 2709
 2710
 2711 002624' 201 03 0 00 000105
 2712 002625' 200 01 0 00 002530*
 2713 002626' 135 02 0 00 013617'
 2714 002627' 316 02 0 00 000003
 2715 002630' 254 00 0 00 002633'
 2716 002631' 260 17 0 00 013627'
 2717 002632' 254 00 0 00 003146'
 2718
 2719
 2720
 2721 002633' 201 03 0 00 000000
 2722 002634' 135 02 0 00 013636'
 2723 002635' 316 02 0 00 000003
 2724 002636' 254 00 0 00 002641'
 2725 002637' 260 17 0 00 013646'
 2726 002640' 254 00 0 00 003146'

TS31AD: \$CALL SNDCMP ; Test for msg/dgm send complete.
 JRST TS31AE ; No failure,continue
 ERROR1 (, Time out waiting for generate datagram send complete,,NNN)
 JRST TS31W

TS31AE: \$CALL DATAVL ; Wait for datagram available flag
 JRST TS31AF ; No failure,continue
 ERROR1 (, Time out waiting for generated datagram,,NNN)
 JRST TS31W

;Read the DATAGRAM

TS31AF: \$CALL READD

;Test response datagram opcode

MOVEI T1,^D69 ;Make correct 69
 MOVE S1,RETBUR
 LDB S2,[POINT 8,(S1),7] ;Put returned opcode in S2
 CAMN S2,T1 ;Is returned status as expected ?
 JRST TS31AH ; Yes
 ERROR1 (3,T1,S2,Response datagram opcode in error,,NNN)
 JRST TS31Y

;Test response datagram status

TS31AH: MOVEI T1,0 ;Set up correct
 LDB S2,[POINT 8,1(S1),15] ; S2=STATUS
 CAMN S2,T1 ;Is returned status as expected ?
 JRST TS31AI ; Yes
 ERROR1 (3,T1,S2,Response datagram status in error,,NNN)
 JRST TS31Y

2727
 2728
 2729
 2730 002641' 201 03 0 00 000000
 2731 002642' 135 02 0 00 013655'
 2732 002643' 316 02 0 00 000003'
 2733 002644' 254 00 0 00 002647'
 2734 002645' 260 17 0 00 013667'
 2735 002646' 254 00 0 00 003146'
 2736
 2737
 2738
 2739 002647' 201 03 0 00 177777
 2740 002650' 135 02 0 00 013711'
 2741 002651' 316 02 0 00 000003'
 2742 002652' 254 00 0 00 002654'
 2743 002653' 260 17 0 00 013724'
 2744
 2745
 2746
 2747 002654' 135 02 0 00 013733'
 2748 002655' 316 02 0 00 000003'
 2749 002656' 254 00 0 00 002660'
 2750 002657' 260 17 0 00 013746'
 2751
 2752
 2753
 2754 002660' 260 17 0 00 002551*
 2755
 2756
 2757 002661' 201 01 0 00 000764
 2758 002662' 260 17 0 00 002557*

;Test act count = 0.

TS31AI: MOVEI T1,0 ;Set up correct
 LDB S2,[POINT 16,1(S1),31] ; s2= act cnt.
 CAMN S2,T1 ; test ACT CNT = EXPECTED.
 JRST TS31AJ ; Yes
 ERROR1 (6,T1,S2,Response datagram account field in error,,NNN)
 JRST TS31Y

;Test first two data bytes

TS31AJ: MOVEI T1,177777 ;Set up correct
 LDB S2,[POINT 16,3(S1),31] ; s2=1ST 2 BYTES OF DATA.
 CAMN S2,T1 ;Are first 2 bytes correct ?
 JRST TS31AK ; Yes
 ERROR1 (6,T1,S2,Response datagram first 2 data bytes in error,,NNN)

;Test second two data bytes

TS31AK: LDB S2,[POINT 16,4(S1),15] ; s2=2ND 2 BYTES OF DATA.
 CAMN S2,T1 ;Are first 2 bytes correct ?
 JRST TS31AL ; Yes
 ERROR1 (6,T1,S2,Response datagram second 2 data bytes in error,,NNN)

;Requeue the buffer

TS31AL: \$CALL REQUED

;Wait

MOVEI S1,^D500
 \$CALL WAITX

2759
 2760
 2761
 2762
 2763
 2764
 2765 002663'
 2766 002663' 201 01 0 00 000000
 2767 002664' 201 02 0 00 013755'
 2768 002665' 260 17 0 00 011372'
 2769
 2770
 2771
 2772 002666' 201 01 0 00 000005
 2773 002667' 202 01 0 00 002575*
 2774 002670' 202 01 0 00 002576*
 2775
 2776 002671' 201 01 0 00 000000
 2777 002672' 202 01 0 00 002600*
 2778 002673' 202 01 0 00 002601*
 2779 002674' 202 01 0 00 002604*
 2780
 2781 002675' 201 01 0 00 000001
 2782 002676' 202 01 0 00 002602*
 2783
 2784 002677' 260 17 0 00 002605*
 2785
 2786
 2787
 2788 002700' 260 17 0 00 002606*
 2789
 2790
 2791
 2792 002701' 260 17 0 00 011242'
 2793 002702' 254 00 0 00 002705'
 2794 002703' 260 17 0 00 013547'
 2795 002704' 254 00 0 00 003147'
 2796
 2797 002705' 260 17 0 00 011317'
 2798 002706' 254 00 0 00 002711'
 2799 002707' 260 17 0 00 013571'
 2800 002710' 254 00 0 00 003147'
 2801
 2802 002711' 260 17 0 00 011341'
 2803 002712' 254 00 0 00 002715'
 2804 002713' 260 17 0 00 013610'
 2805 002714' 254 00 0 00 003147'
 2806
 2807
 2808
 2809 002715' 260 17 0 00 002623*

 * TST31B - DATAGRAM TEST. THIS TEST WILL SEND A DATAGRAM REQUEST OF
 DATA=BYTE PAIR, LENGTH=4, DELAY=0, REPEAT COUNT=0.

TST31B:
 TS31B: MOVEI S1,0 ;Load 0 for no message output
 MOVEI S2,[ASCIZ/Test 31 Part B - Datagram Test (Data=byte pairs)/]
 \$CALL TRNODE ;Store that message please

;Build generate datagram ctp packet

MOVEI S1,5
 MOVEM S1,OPCODE ; OPCODE = GEN DATAGRAM
 MOVEM S1,GENLEN ; GEN LENGTH=4.
 MOVEI S1,0
 MOVEM S1,DELAY ; DELAY = 0.
 MOVEM S1,RCOUNT ; REPEAT COUNT = 0.
 MOVEM S1,GENCONST ; GENCONST=0.

MOVEI S1,1
 MOVEM S1,GENFUNCT ; GENFUNCT =1/GENFBPAIR

\$CALL BLDCTP

;Send the datagram

\$CALL SNDDAT

;Wait for msg/dgm send complete and datagram available

\$CALL TEVPND ; Test for event pending.
 JRST TS31BD ; No failure,continue
 ERROR1 (,,Time out waiting for event pending after generate datagram sent,,NNN)
 JRST TS31W

TS31BD: \$CALL SNDCMP ; Test for msg/dgm send complete.
 JRST TS31BE ; No failure,continue
 ERROR1 (,,Time out waiting for generate datagram send complete,,NNN)
 JRST TS31W

TS31BE: \$CALL DATAVL ; Wait for datagram available flag
 JRST TS31BF ; No failure,continue
 ERROR1 (,,Time out waiting for generated datagram,,NNN)
 JRST TS31W

;Read the DATAGRAM

TS31BF: \$CALL READD

2810
2811
2812
2813 002716' 201 03 0 00 000105
2814 002717' 200 01 0 00 002625*
2815 002720' 135 02 0 00 013617'
2816 002721' 316 02 0 00 000003
2817 002722' 254 00 0 00 002725'
2818 002723' 260 17 0 00 013627'
2819 002724' 254 00 0 00 003146'
2820
2821
2822
2823 002725' 201 03 0 00 000000
2824 002726' 135 02 0 00 013636'
2825 002727' 316 02 0 00 000003
2826 002730' 254 00 0 00 002733'
2827 002731' 260 17 0 00 013646'
2828 002732' 254 00 0 00 003146'
2829
2830
2831
2832 002733' 201 03 0 00 000000
2833 002734' 135 02 0 00 013655'
2834 002735' 316 02 0 00 000003
2835 002736' 254 00 0 00 002741'
2836 002737' 260 17 0 00 013667'
2837 002740' 254 00 0 00 003146'
2838
2839
2840
2841 002741' 201 03 0 00 000000
2842 002742' 135 02 0 00 013711'
2843 002743' 316 02 0 00 000003
2844 002744' 254 00 0 00 002746'
2845 002745' 260 17 0 00 013724'
2846
2847
2848
2849 002746' 201 03 0 00 000400
2850 002747' 135 02 0 00 013733'
2851 002750' 316 02 0 00 000003
2852 002751' 254 00 0 00 002753'
2853 002752' 260 17 0 00 013746'
2854
2855
2856
2857 002753' 260 17 0 00 002660*
2858
2859
2860 002754' 201 01 0 00 000764
2861 002755' 260 17 0 00 002662*

;Test response datagram opcode

```

MOVEI T1,^D69 ;Make correct 69
MOVE S1,RETBUF
LDB S2,[POINT 8,(S1),7] ;Put returned opcode in S2
CAMN S2,T1 ;Is returned status as expected ?
JRST TS31BH ; Yes
ERROR1 (3,T1,S2,Response datagram opcode in error,,NNN)
JRST TS31Y

```

;Test response datagram status

```

TS31BH: MOVEI T1,0 ;Set up correct
LDB S2,[POINT 8,1(S1),15] ; S2=STATUS
CAMN S2,T1 ;Is returned status as expected ?
JRST TS31BI ; Yes
ERROR1 (3,T1,S2,Response datagram status in error,,NNN)
JRST TS31Y

```

;Test act count = 0.

```

TS31BI: MOVEI T1,0 ;Set up correct
LDB S2,[POINT 16,1(S1),31] ; s2= act cnt.
CAMN S2,T1 ; test ACT CNT = EXPECTED.
JRST TS31BJ ; Yes
ERROR1 (6,T1,S2,Response datagram account field in error,,NNN)
JRST TS31Y

```

;Test first two data bytes

```

TS31BJ: MOVEI T1,0 ;Set up correct
LDB S2,[POINT 16,3(S1),31] ; s2=1ST 2 BYTES OF DATA.
CAMN S2,T1 ;Are first 2 bytes correct ?
JRST TS31BK ; Yes
ERROR1 (6,T1,S2,Response datagram first 2 data bytes in error,,NNN)

```

;Test second two data bytes

```

TS31BK: MOVEI T1,400 ;Set up correct
LDB S2,[POINT 16,4(S1),15] ; s2=2ND 2 BYTES OF DATA.
CAMN S2,T1 ;Are first 2 bytes correct ?
JRST TS31BL ; Yes
ERROR1 (6,T1,S2,Response datagram second 2 data bytes in error,,NNN)

```

;Requeue the buffer

TS31BL: \$CALL REQUED

;Wait

```

MOVEI S1,^D500
$CALL WAITX

```

```

2862
2863
2864
2865
2866
2867
2868 002756'
2869 002756' 201 01 0 00 000000
2870 002757' 201 02 0 00 013767'
2871 002760' 260 17 0 00 011372'
2872
2873
2874
2875 002761' 201 01 0 00 000005
2876 002762' 202 01 0 00 002667*
2877 002763' 202 01 0 00 002670*
2878
2879 002764' 201 01 0 00 000002
2880 002765' 202 01 0 00 002676*
2881
2882 002766' 201 01 0 00 000000
2883 002767' 202 01 0 00 002672*
2884 002770' 202 01 0 00 002673*
2885 002771' 202 01 0 00 002674*
2886
2887
2888
2889 002772' 402 00 0 00 011431'
2890 002773' 200 01 0 00 002340*
2891 002774' 202 01 0 00 011417'
2892 002775' 242 01 0 00 000010
2893 002776' 436 01 0 00 011417'
2894
2895 002777' 260 17 0 00 002677*
2896
2897
2898
2899 003000' 260 17 0 00 002700*
2900
2901
2902
2903 003001' 260 17 0 00 011242'
2904 003002' 254 00 0 00 003005'
2905 003003' 260 17 0 00 013547'
2906 003004' 254 00 0 00 003147'
2907
2908 003005' 260 17 0 00 011317'
2909 003006' 254 00 0 00 003011'
2910 003007' 260 17 0 00 013571'
2911 003010' 254 00 0 00 003147'
2912
2913 003011' 260 17 0 00 011341'
2914 003012' 254 00 0 00 003015'
2915 003013' 260 17 0 00 013610'
2916 003014' 254 00 0 00 003147'

```

```

*****
;* TST31C - DATAGRAM TEST. THIS TEST WILL SEND A DATAGRAM REQUEST OF
;* DATA=NODE, LENGTH=4, DELAY=0, REPEAT COUNT=0.
*****

```

```

TST31C:
TS31C:  MOVEI    S1,0           ;Load 0 for no message output
        MOVEI    S2,[ASCIZ/Test 31 Part C - Datagram Test (Data=node number)/]
        $CALL    TRNODE       ;Store that message please

```

```

;Build generate datagram ctp packet

```

```

        MOVEI    S1,5
        MOVEM    S1,OPCODE     ; OPCODE = GEN DATAGRAM
        MOVEM    S1,GENLEN     ; GEN LENGTH=4.

        MOVEI    S1,2
        MOVEM    S1,GENFUN     ; GENFUNCT =2/GENFNODE

        MOVEI    S1,0
        MOVEM    S1,DELAY     ; DELAY = 0.
        MOVEM    S1,RCOUNT    ; REPEAT COUNT = 0.
        MOVEM    S1,GENCONST  ; GENCONST=0.

```

```

;Create node comparison value.

```

```

        SETZM    NODCMP       ; Clear comparator word
        MOVE     S1,NODEN     ; Node number to S1
        MOVEM    S1,NODDTA    ; Node number to nodcmp.
        LSH      S1,8         ; Shift over 1st number.
        IORM     S1,NODDTA    ; Plop in 2nd node number.

```

```

        $CALL    BLDCTP       ;Go build the ctp packet

```

```

;Send the DATAGRAM

```

```

        $CALL    SNDDAT

```

```

;Wait for msg/dgm send complete and datagram available

```

```

        $CALL    TEVPND       ; Test for event pending.
        JRST     TS31CD       ; No failure,continue
        ERROR1   (,,Timed out waiting for event pending after generate datagram sent,,NNN)
        JRST     TS31W

```

```

TS31CD: $CALL    SNDCMP       ; Test for msg/dgm send complete.
        JRST     TS31CE       ; No failure,continue
        ERROR1   (,,Time out waiting for generate datagram send complete,,NNN)
        JRST     TS31W

```

```

TS31CE: $CALL    DATAVL     ; Wait for datagram available flag
        JRST     TS31CF       ; No failure,continue
        ERROR1   (,,Time out waiting for generated datagram,,NNN)
        JRST     TS31W

```


2917
 2918
 2919
 2920 003015' 260 17 0 00 002715*
 2921
 2922
 2923
 2924 003016' 201 03 0 00 000105
 2925 003017' 200 01 0 00 002717*
 2926 003020' 135 02 0 00 013617'
 2927 003021' 316 02 0 00 000003
 2928 003022' 254 00 0 00 003025'
 2929 003023' 260 17 0 00 013627'
 2930 003024' 254 00 0 00 003146'
 2931
 2932
 2933
 2934 003025' 201 03 0 00 000000
 2935 003026' 135 02 0 00 013636'
 2936 003027' 316 02 0 00 000003
 2937 003030' 254 00 0 00 003033'
 2938 003031' 260 17 0 00 013646'
 2939 003032' 254 00 0 00 003146'
 2940
 2941
 2942
 2943 003033' 201 03 0 00 000000
 2944 003034' 135 02 0 00 013655'
 2945 003035' 316 02 0 00 000003
 2946 003036' 254 00 0 00 003041'
 2947 003037' 260 17 0 00 013667'
 2948 003040' 254 00 0 00 003146'
 2949
 2950
 2951
 2952 003041' 135 02 0 00 013711'
 2953 003042' 316 02 0 00 011417'
 2954 003043' 254 00 0 00 003045'
 2955 003044' 260 17 0 00 014001'
 2956
 2957
 2958
 2959 003045' 135 02 0 00 013733'
 2960 003046' 316 02 0 00 011417'
 2961 003047' 254 00 0 00 003051'
 2962 003050' 260 17 0 00 014010'
 2963
 2964
 2965
 2966 003051' 260 17 0 00 002753*
 2967
 2968
 2969 003052' 201 01 0 00 000764
 2970 003053' 260 17 0 00 002755*

;Read the DATAGRAM

TS31CF: \$CALL READD

;Test response datagram opcode

```

      MOVEI T1,^D69           ;Make correct 69
      MOVE  S1,RETBUF
      LDB   S2,[POINT 8,(S1),7] ;Put returned opcode in S2
      CAMN  S2,T1             ;Is returned status as expected ?
      JRST  TS31CH            ; Yes
      ERROR1 (3,T1,S2,Response datagram opcode in error,,NNN)
      JRST  TST31Y
  
```

;Test response datagram status

```

TS31CH: MOVEI T1,0           ;Set up correct
      LDB   S2,[POINT 8,1(S1),15] ; S2=STATUS
      CAMN  S2,T1             ;Is returned status as expected ?
      JRST  TS31CI            ; Yes
      ERROR1 (3,T1,S2,Response datagram status in error,,NNN)
      JRST  TST31Y
  
```

;Test act count = 0.

```

TS31CI: MOVEI T1,0           ;Set up correct
      LDB   S2,[POINT 16,1(S1),31] ; s2= act cnt.
      CAMN  S2,T1             ; test ACT CNT = EXPECTED.
      JRST  TS31CJ            ; Yes
      ERROR1 (6,T1,S2,Response datagram account field in error,,NNN)
      JRST  TST31Y
  
```

;Test first two data bytes

```

TS31CJ: LDB   S2,[POINT 16,3(S1),31] ; s2=1ST 2 BYTES OF DATA.
      CAMN  S2,NODDTA         ;Are first 2 bytes correct ?
      JRST  TS31CK            ; Yes
      ERROR1 (6,NODDTA,S2,Response datagram first 2 data bytes in error,,NNN)
  
```

;Test second two data bytes

```

TS31CK: LDB   S2,[POINT 16,4(S1),15] ; s2=2ND 2 BYTES OF DATA.
      CAMN  S2,NODDTA         ;Are first 2 bytes correct ?
      JRST  TS31CL            ; Yes
      ERROR1 (6,NODDTA,S2,Response datagram second 2 data bytes in error,,NNN)
  
```

;Requeue the buffer

TS31CL: \$CALL REQUED

;Wait

```

      MOVEI S1,^D50
      $CALL WAITX
  
```


2971
 2972
 2973
 2974
 2975
 2976
 2977 003054' 201 01 0 00 000000
 2978 003054' 201 01 0 00 000000
 2979 003055' 201 02 0 00 014017'
 2980 003056' 260 17 0 00 011372'
 2981
 2982
 2983
 2984 003057' 201 01 0 00 000005
 2985 003060' 202 01 0 00 002762*
 2986 003061' 202 01 0 00 002763*
 2987 003062' 201 01 0 00 000000
 2988 003063' 202 01 0 00 002767*
 2989 003064' 202 01 0 00 002770*
 2990 003065' 202 01 0 00 002771*
 2991 003066' 201 01 0 00 000003
 2992 003067' 202 01 0 00 002765*
 2993 003070' 200 01 0 00 012474'
 2994 003071' 202 01 0 00 011433'
 2995 003072' 260 17 0 00 002777*
 2996
 2997
 2998
 2999
 3000
 3001
 3002 003073' 260 17 0 00 003000*
 3003
 3004
 3005
 3006 003074' 260 17 0 00 011242'
 3007 003075' 254 00 0 00 003100'
 3008 003076' 260 17 0 00 013547'
 3009 003077' 254 00 0 00 003147'
 3010
 3011 003100' 260 17 0 00 011317'
 3012 003101' 254 00 0 00 003104'
 3013 003102' 260 17 0 00 013571'
 3014 003103' 254 00 0 00 003147'
 3015
 3016 003104' 260 17 0 00 011341'
 3017 003105' 254 00 0 00 003110'
 3018 003106' 260 17 0 00 013610'
 3019 003107' 254 00 0 00 003147'

 ; * TS31D - DATAGRAM TEST. THIS TEST WILL SEND A DATAGRAM REQUEST OF
 ; DATA=IMAGE DATA, LENGTH=4, DELAY=0, REPEAT COUNT=0.
 ; *****

TST31D:
 TS31D: MOVEI S1,0 ;Load 0 for no message output
 MOVEI S2,[ASCIZ/Test 31 Part D - Datagram Test (Data=image data)/]
 \$CALL TRNODE ;Store that message please

;Build generate datagram ctp packet

MOVEI S1,5
 MOVEM S1,OPCODE ; OPCODE = GEN DATAGRAM.
 MOVEM S1,GENLEN ; GEN LENGTH=4.
 MOVEI S1,0
 MOVEM S1,DELAY ; DELAY = 0.
 MOVEM S1,RCOUNT ; REPEAT COUNT = 0.
 MOVEM S1,GNCNST ; GENCONST=0.
 MOVEI S1,3
 MOVEM S1,GENFUN ; GENFUNCT =3/GENF IMAGE
 MOVE S1,[BYTE(8)377,376,375,374] ; IMAGE=377,376,375,374.
 MOVEM S1,IMAGDT
 \$CALL BLDCTP

;Build the comparison words for image data

;Send the DATAGRAM

\$CALL SNDDAT

;Wait for msg/dgm send complete and datagram available

\$CALL TEVPND ; Test for event pending.
 JRST TS31DD ; No failure,continue
 ERROR1 (,Timed out waiting for event pending after generate datagram sent,,NNN)
 JRST TS31W

TS31DD: \$CALL SNDCMP ; Test for msg/dgm send complete.
 JRST TS31DE ; No failure,continue
 ERROR1 (,Time out waiting for generate datagram send complete,,NNN)
 JRST TS31W

TS31DE: \$CALL DATAVL ; Wait for datagram available flag
 JRST TS31DF ; No failure,continue
 ERROR1 (,Time out waiting for generated datagram,,NNN)
 JRST TS31W

3020
 3021
 3022
 3023 003110' 260 17 0 00 003015*
 3024
 3025
 3026
 3027 003111' 201 03 0 00 000105
 3028 003112' 200 01 0 00 003017*
 3029 003113' 135 02 0 00 013617'
 3030 003114' 316 02 0 00 000003
 3031 003115' 254 00 0 00 003120'
 3032 003116' 260 17 0 00 013627'
 3033 003117' 254 00 0 00 003146'
 3034
 3035
 3036
 3037 003120' 201 03 0 00 000000
 3038 003121' 135 02 0 00 013636'
 3039 003122' 316 02 0 00 000003
 3040 003123' 254 00 0 00 003126'
 3041 003124' 260 17 0 00 013646'
 3042 003125' 254 00 0 00 003146'
 3043
 3044
 3045
 3046 003126' 201 03 0 00 000000
 3047 003127' 135 02 0 00 013655'
 3048 003130' 316 02 0 00 000003
 3049 003131' 254 00 0 00 003137'
 3050 003132' 260 17 0 00 013667'
 3051 003133' 254 00 0 00 003146'

;Read the DATAGRAM

TS31DF: \$CALL READD

;Test response datagram opcode

```

      MOVEI T1,^D69          ;Make correct 69
      MOVE  S1,RETBUF
      LDB   S2,[POINT 8,(S1),7] ;Put returned opcode in S2
      CAMN  S2,T1             ;Is returned status as expected ?
      JRST  TS31DH            ; Yes
      ERROR1 (3,T1,S2,Response datagram opcode in error,,NNN)
      JRST  TST31Y
  
```

;Test response datagram status

```

TS31DH: MOVEI T1,0          ;Set up correct
      LDB   S2,[POINT 8,1(S1),15] ; S2=STATUS
      CAMN  S2,T1             ;Is returned status as expected ?
      JRST  TS31DI            ; Yes
      ERROR1 (3,T1,S2,Response datagram status in error,,NNN)
      JRST  TST31Y
  
```

;Test act count = 0.

```

TS31DI: MOVEI T1,0          ;Set up correct
      LDB   S2,[POINT 16,1(S1),31] ; s2= act cnt.
      CAMN  S2,T1             ; test ACT CNT = EXPECTED.
      JRST  TS31DJ            ; Yes
      ERROR1 (6,T1,S2,Response datagram account field in error,,NNN)
      JRST  TST31Y
  
```

3052
 3053
 3054
 3055 003134' 201 03 0 00 176375
 3056 003135' 135 02 0 00 013711'
 3057 003136' 316 02 0 00 000003
 3058 003137' 254 00 0 00 003141'
 3059 003140' 260 17 0 00 013724'
 3060
 3061
 3062
 3063 003141' 201 03 0 00 177377
 3064 003142' 135 02 0 00 013733'
 3065 003143' 316 02 0 00 000003
 3066 003144' 254 00 0 00 003146'
 3067 003145' 260 17 0 00 013746'
 3068
 3069
 3070
 3071 003146' 260 17 0 00 003051*
 3072
 3073
 3074
 3075 003147' 260 17 0 00 002552*
 3076
 3077
 3078
 3079 003150' 260 17 0 00 011306'
 3080 003151' 254 00 0 00 003153'
 3081 003152' 260 17 0 00 011650'
 3082
 3083
 3084
 3085 003153' 201 01 0 00 000764
 3086 003154' 260 17 0 00 003053*
 3087 003155' 263 17 0 00 000000

;Test first two data bytes

TS31DJ: MOVEI T1,<BYTE(8)374,375> -^D20 ;Set up 374,375 in T1
 LDB S2,[POINT 16,3(S1),31] ; s2=1ST 2 BYTES OF DATA.
 CAMN S2,T1 ;Are first 2 bytes correct ?
 JRST TS31DK ; Yes
 ERROR1 (6,T1,S2,Response datagram first 2 data bytes in error,,NNN)

;Test second two data bytes

TS31DK: MOVEI T1,<BYTE(8)376,377> -^D20 ;Set up correct 376,377
 LDB S2,[POINT 16,4(S1),T5] ; s2=2ND 2 BYTES OF DATA.
 CAMN S2,T1 ;Are first 2 bytes correct ?
 JRST TS31Y ; Yes
 ERROR1 (6,T1,S2,Response datagram second 2 data bytes in error,,NNN)

;Requeue the buffer

TS31Y: \$CALL REQUED

;Disconnect with selected node

TST31W: \$CALL DODIS ;Disconnect with the selected node

;Wait for connection closed

\$CALL CONCLO
 JRST TST31X
 ERROR1 (,,,Connection closed not in disconnect status,,NNN)

;Wait

TST31X: MOVEI S1,^D500
 \$CALL WAITX
 \$RET

SUBTTL TST32 - Datagram 4 Bytes Each Of 4 Data Types (Repeat=3)

 ;* TST32 - DATAGRAM TEST. THIS TEST WILL SEND A DATAGRAM REQUEST OF
 ; DATA=376, LENGTH=4, DELAY=0, REPEAT COUNT=3.
 ;*****

TST32:

T32A: MOVEI S1,ASCIZ/TST32/] ;Load test name
 MOVEI S2,ASCIZ/Test 32 Part A - Datagram Test (Data=376, Repeat=3)/]
 \$CALL TRNODE ;Print node number if trace switch set

;Connect with selected node and verify connection=open.

\$CALL CONNCT
 JRST TST32A ; JUMP IF CONNECTION OK.
 ERROR1 (,,,Timed out waiting for event pending flag during connection attempt,,NNN)
 JRST TST32X
 ERROR1 (,,,Event pending was not connection accepted during connection attempt,,NNN
)
 JRST TST32X
 ERROR1 (,,,Connection not open after connection accepted,,NNN)
 JRST TST32X

;Build generate datagram ctp packet

TST32A: MOVEI S1,5
 MOV S1,OPCODE ; OPCODE = GEN DATAGRAM
 MOVEI S1,GENLEN ; GEN LENGTH=4.
 MOVEI S1,0
 MOVEM S1,GENFUN ; GENFUNCT =0/GENFILL.
 MOVEM S1,DELAY ; DELAY = 0
 MOVEI S1,376
 MOVEM S1,GENCONST ; GENCONST=376.
 MOVEI S1,3
 MOVEM S1,RCOUNT ; REPEAT COUNT = 3.
 \$CALL BLDCTP

;Send the datagram

\$CALL SNDDAT

3088
 3089
 3090
 3091
 3092
 3093
 3094
 3095 003156' 201 01 0 00 014031'
 3096 003156' 201 01 0 00 014031'
 3097 003157' 201 02 0 00 014033'
 3098 003160' 260 17 0 00 011372'
 3099
 3100
 3101
 3102 003161' 260 17 0 00 011134'
 3103 003162' 254 00 0 00 003171'
 3104
 3105 003163' 260 17 0 00 011562'
 3106 003164' 254 00 0 00 004447'
 3107
 3108 003165' 260 17 0 00 011607'
 3109 003166' 254 00 0 00 004447'
 3110 003167' 260 17 0 00 011630'
 3111 003170' 254 00 0 00 004447'
 3112
 3113
 3114
 3115 003171' 201 01 0 00 000005
 3116 003172' 202 01 0 00 003060*
 3117 003173' 202 01 0 00 003061*
 3118 003174' 201 01 0 00 000000
 3119 003175' 202 01 0 00 003067*
 3120 003176' 202 01 0 00 003063*
 3121 003177' 201 01 0 00 000376
 3122 003200' 202 01 0 00 003065*
 3123 003201' 201 01 0 00 000003
 3124 003202' 202 01 0 00 003064*
 3125 003203' 260 17 0 00 003072*
 3126
 3127
 3128
 3129 003204' 260 17 0 00 003073*


```

3130
3131
3132
3133 003205' 260 17 0 00 011242'
3134 003206' 254 00 0 00 003211'
3135 003207' 260 17 0 00 013547'
3136 003210' 254 00 0 00 004442'
3137
3138 003211' 260 17 0 00 011317'
3139 003212' 254 00 0 00 003215'
3140 003213' 260 17 0 00 013571'
3141 003214' 254 00 0 00 004442'
3142
3143 003215' 260 17 0 00 011341'
3144 003216' 254 00 0 00 003221'
3145 003217' 260 17 0 00 013610'
3146 003220' 254 00 0 00 004442'
3147
3148
3149
3150 003221' 260 17 0 00 003110*
3151
3152
3153
3154 003222' 201 03 0 00 000105
3155 003223' 200 01 0 00 003112*
3156 003224' 135 02 0 00 013617'
3157 003225' 316 02 0 00 000003
3158 003226' 254 00 0 00 003231'
3159 003227' 260 17 0 00 013627'
3160 003230' 254 00 0 00 004442'
3161
3162
3163
3164 003231' 201 03 0 00 000000
3165 003232' 135 02 0 00 013636'
3166 003233' 316 02 0 00 000003
3167 003234' 254 00 0 00 003237'
3168 003235' 260 17 0 00 013646'
3169 003236' 254 00 0 00 004442'
3170
3171
3172
3173 003237' 201 03 0 00 000000
3174 003240' 135 02 0 00 013655'
3175 003241' 316 02 0 00 000003
3176 003242' 254 00 0 00 003245'
3177 003243' 260 17 0 00 013667'
3178 003244' 254 00 0 00 004442'
  
```

;Wait for msg/dgm send complete and datagram available

```

$CALL TEVPND ; TEST FOR EVENT PENDING.
JRST T32S1 ; NO FAILURE,CONTINUE
ERROR1 (,Time out waiting for event pending after generate datagram sent,,NNN)
JRST T32Y
  
```

```

T32S1: $CALL SNDCMP ; Test for msg/dgm send complete.
JRST T32S2 ; No failure, continue
ERROR1 (,Time out waiting for generate datagram send complete,,NNN)
JRST T32Y
  
```

```

T32S2: $CALL DATAVL ; Wait for datagram available flag
JRST T32T1 ; No failure, continue
ERROR1 (,Time out waiting for generated datagram,,NNN)
JRST T32Y
  
```

;Read the datagram

```
T32T1: $CALL READD
```

;Test response datagram opcode

```

MOVEI T1,^D69 ;Make correct 69
MOVE S1,RETBUF
LDB S2,[POINT 8,(S1),7] ;Put returned opcode in S2
CAMN S2,T1 ;Is returned status as expected ?
JRST T32AD1 ; Yes
ERROR1 (3,T1,S2,Response datagram opcode in error,,NNN)
JRST T32Y
  
```

;Test response datagram status

```

T32AD1: MOVEI T1,0 ;Set up correct
LDB S2,[POINT 8,1(S1),15] ; S2=STATUS
CAMN S2,T1 ;Is returned status as expected ?
JRST T32AD2 ; Yes
ERROR1 (3,T1,S2,Response datagram status in error,,NNN)
JRST T32Y
  
```

;Test act count = 0.

```

T32AD2: MOVEI T1,0 ;Set up correct
LDB S2,[POINT 16,1(S1),31] ; s2= act cnt.
CAMN S2,T1 ; test ACT CNT = EXPECTED.
JRST T32AD3 ; Yes
ERROR1 (6,T1,S2,Response datagram account field in error,,NNN)
JRST T32Y
  
```


3179
 3180
 3181
 3182 003245' 201 03 0 00 177376
 3183 003246' 135 02 0 00 013711'
 3184 003247' 316 02 0 00 000003
 3185 003250' 254 00 0 00 003253'
 3186 003251' 260 17 0 00 013724'
 3187 003252' 254 00 0 00 004442'
 3188
 3189
 3190
 3191 003253' 135 02 0 00 013733'
 3192 003254' 316 02 0 00 000003
 3193 003255' 254 00 0 00 003260'
 3194 003256' 260 17 0 00 013746'
 3195 003257' 254 00 0 00 004442'
 3196
 3197
 3198
 3199 003260' 260 17 0 00 003146*
 3200
 3201
 3202
 3203 003261' 260 17 0 00 011220'
 3204 003262' 254 00 0 00 003265'
 3205 003263' 260 17 0 00 013610'
 3206 003264' 254 00 0 00 004442'
 3207
 3208
 3209
 3210 003265' 260 17 0 00 003221*
 3211
 3212
 3213
 3214 003266' 201 03 0 00 000105
 3215 003267' 200 01 0 00 003223*
 3216 003270' 135 02 0 00 013617'
 3217 003271' 316 02 0 00 000003
 3218 003272' 254 00 0 00 003275'
 3219 003273' 260 17 0 00 013627'
 3220 003274' 254 00 0 00 004442'
 3221
 3222
 3223
 3224 003275' 201 03 0 00 000000
 3225 003276' 135 02 0 00 013616'
 3226 003277' 316 02 0 00 000003
 3227 003300' 254 00 0 00 003303'
 3228 003301' 260 17 0 00 013646'
 3229 003302' 254 00 0 00 004442'

;Test first two data bytes

T32AD3: MOVEI T1,177376 ;Set up correct
 LDB S2,[POINT 16,3(S1),31] ; s2=1ST 2 BYTES OF DATA.
 CAMN S2,T1 ;Are first 2 bytes correct ?
 JRST T32AD4 ; Yes
 ERROR1 (6,T1,S2,Response datagram first 2 data bytes in error,,NNN)
 JRST TST32Y

;Test second two data bytes

T32AD4: LDB S2,[POINT 16,4(S1),15] ; s2=2ND 2 BYTES OF DATA.
 CAMN S2,T1 ;Are first 2 bytes correct ?
 JRST T32AD5 ; Yes
 ERROR1 (6,T1,S2,Response datagram second 2 data bytes in error,,NNN)
 JRST TST32Y

;Requeue the buffer

T32AD5: \$CALL REQUED

;Wait for datagram available

\$CALL WAITDA ; Wait for datagram available flag
 JRST T32AE1 ; No failure, continue
 ERROR1 (,,Time out waiting for generated datagram,,NNN)
 JRST TST32Y

;Read the datagram

T32AE1: \$CALL READD

;Test response datagram opcode

MOVEI T1,^D69 ;Make correct 69
 MOVE S1,RETBUF
 LDB S2,[POINT 8,(S1),7] ;Put returned opcode in S2
 CAMN S2,T1 ;Is returned status as expected ?
 JRST T32AE2 ; Yes
 ERROR1 (3,T1,S2,Response datagram opcode in error,,NNN)
 JRST TST32Y

;Test response datagram status

T32AE2: MOVEI T1,0 ;Set up correct
 LDB S2,[POINT 8,1(S1),15] ; S2=STATUS
 CAMN S2,T1 ;Is returned status as expected ?
 JRST T32AE3 ; Yes
 ERROR1 (3,T1,S2,Response datagram status in error,,NNN)
 JRST TST32Y

```

3230
3231
3232
3233 003303' 201 03 0 00 000400
3234 003304' 135 02 0 00 013655'
3235 003305' 316 02 0 00 000003
3236 003306' 254 00 0 00 003311'
3237 003307' 260 17 0 00 013667'
3238 003310' 254 00 0 00 004442'
3239
3240
3241
3242 003311' 201 03 0 00 177376
3243 003312' 135 02 0 00 013711'
3244 003313' 316 02 0 00 000003
3245 003314' 254 00 0 00 003317'
3246 003315' 260 17 0 00 013724'
3247 003316' 254 00 0 00 004442'
3248
3249
3250
3251 003317' 135 02 0 00 013733'
3252 003320' 316 02 0 00 000003
3253 003321' 254 00 0 00 003324'
3254 003322' 260 17 0 00 013746'
3255 003323' 254 00 0 00 004442'
3256
3257
3258
3259 003324' 260 17 0 00 002446*
3260
3261
3262
3263 003325' 260 17 0 00 011220'
3264 003326' 254 00 0 00 003331'
3265 003327' 260 17 0 00 013610'
3266 003330' 254 00 0 00 004442'
3267
3268
3269
3270 003331' 260 17 0 00 003265*
3271
3272
3273
3274 003332' 201 03 0 00 000105
3275 003333' 200 01 0 00 003267*
3276 003334' 135 02 0 00 013617'
3277 003335' 316 02 0 00 000003
3278 003336' 254 00 0 00 003341'
3279 003337' 260 17 0 00 013627'
3280 003340' 254 00 0 00 004442'

```

;Test act count = 1.

```

T32AE3: MOVEI T1,1*^D256 ;Set up correct
LDB S2,[POINT 16,1(S1),31] ; s2= act cnt.
CAMN S2,T1 ; test ACT CNT = EXPECTED.
JRST T32AE4 ; Yes
ERROR1 (6,T1,S2,Response datagram account field in error,,NNN)
JRST TST32Y

```

;Test first two data bytes

```

T32AE4: MOVEI T1,177376 ;Set up correct
LDB S2,[POINT 16,3(S1),31] ; s2=1ST 2 BYTES OF DATA.
CAMN S2,T1 ;Are first 2 bytes correct ?
JRST T32AE5 ; Yes
ERROR1 (6,T1,S2,Response datagram first 2 data bytes in error,,NNN)
JRST TST32Y

```

;Test second two data bytes

```

T32AE5: LDB S2,[POINT 16,4(S1),15] ; s2=2ND 2 BYTES OF DATA.
CAMN S2,T1 ;Are first 2 bytes correct ?
JRST T32AE6 ; Yes
ERROR1 (6,T1,S2,Response datagram second 2 data bytes in error,,NNN)
JRST TST32Y

```

;Requeue the buffer

```

T32AE6: $CALL REQUEM

```

;Wait for datagram available

```

$CALL WAITDA ; Wait for datagram available flag
JRST T32AF1 ; No failure, continue
ERROR1 (,Time out waiting for generated datagram,,NNN)
JRST TST32Y

```

;Read the datagram

```

T32AF1: $CALL READD

```

;Test response datagram opcode

```

MOVEI T1,^D69 ;Make correct 69
MOVE S1,RETBUF
LDB S2,[POINT 8,(S1),7] ;Put returned opcode in S2
CAMN S2,T1 ;Is returned status as expected ?
JRST T32AF2 ; Yes
ERROR1 (3,T1,S2,Response datagram opcode in error,,NNN)
JRST TST32Y

```

3281
 3282
 3283
 3284 003341' 201 03 0 00 000000
 3285 003342' 135 02 0 00 013636'
 3286 003343' 316 02 0 00 000003
 3287 003344' 254 00 0 00 003347'
 3288 003345' 260 17 0 00 013646'
 3289 003346' 254 00 0 00 004442'
 3290
 3291
 3292
 3293 003347' 201 03 0 00 001000
 3294 003350' 135 02 0 00 013655'
 3295 003351' 316 02 0 00 000003
 3296 003352' 254 00 0 00 003355'
 3297 003353' 260 17 0 00 013667'
 3298 003354' 254 00 0 00 004442'
 3299
 3300
 3301
 3302 003355' 201 03 0 00 177376
 3303 003356' 135 02 0 00 013711'
 3304 003357' 316 02 0 00 000003
 3305 003360' 254 00 0 00 003363'
 3306 003361' 260 17 0 00 013724'
 3307 003362' 254 00 0 00 004442'
 3308
 3309
 3310
 3311 003363' 135 02 0 00 013733'
 3312 003364' 316 02 0 00 000003
 3313 003365' 254 00 0 00 003370'
 3314 003366' 260 17 0 00 013746'
 3315 003367' 254 00 0 00 004442'
 3316
 3317
 3318
 3319 003370' 260 17 0 00 003260*
 3320
 3321
 3322
 3323 003371' 260 17 0 00 011220'
 3324 003372' 254 00 0 00 003375'
 3325 003373' 260 17 0 00 013610'
 3326 003374' 254 00 0 00 004442'
 3327
 3328
 3329
 3330 003375' 260 17 0 00 003331*

;Test response datagram status

T32AF2: MOVEI T1,0 ;Set up correct
 LDB S2,[POINT 8,1(S1),15] ; S2=STATUS
 CAMN S2,T1 ;Is returned status as expected ?
 JRST T32AF3 ; Yes
 ERROR1 (3,T1,S2,Response datagram status in error,,NNN)
 JRST TST32Y

;Test act count = 2.

T32AF3: MOVEI T1,2*D256 ;Set up correct
 LDB S2,[POINT 16,1(S1),31] ; s2= act cnt.
 CAMN S2,T1 ; test ACT CNT = EXPECTED.
 JRST T32AF4 ; Yes
 ERROR1 (6,T1,S2,Response datagram account field in error,,NNN)
 JRST TST32Y

;Test first two data bytes

T32AF4: MOVEI T1,177376 ;Set up correct
 LDB S2,[POINT 16,3(S1),31] ; s2=1ST 2 BYTES OF DATA.
 CAMN S2,T1 ;Are first 2 bytes correct ?
 JRST T32AF5 ; Yes
 ERROR1 (6,T1,S2,Response datagram first 2 data bytes in error,,NNN)
 JRST TST32Y

;Test second two data bytes

T32AF5: LDB S2,[POINT 16,4(S1),15] ; s2=2ND 2 BYTES OF DATA.
 CAMN S2,T1 ;Are first 2 bytes correct ?
 JRST T32AF6 ; Yes
 ERROR1 (6,T1,S2,Response datagram second 2 data bytes in error,,NNN)
 JRST TST32Y

;Requeue the buffer

T32AF6: \$CALL REQUED

;Wait for datagram available

\$CALL WAITDA ; Wait for datagram available flag
 JRST T32AG1 ; No failure, continue
 ERROR1 (,,Time out waiting for generated datagram,,NNN)
 JRST TST32Y

;Read the datagram

T32AG1: \$CALL READD

3331
 3332
 3333
 3334 003376' 201 03 0 00 000105
 3335 003377' 200 01 0 00 003333*
 3336 003400' 135 02 0 00 013617'
 3337 003401' 316 02 0 00 000003
 3338 003402' 254 00 0 00 003405'
 3339 003403' 260 17 0 00 013627'
 3340 003404' 254 00 0 00 004442'
 3341
 3342
 3343
 3344 003405' 201 03 0 00 000000
 3345 003406' 135 02 0 00 013636'
 3346 003407' 316 02 0 00 000003
 3347 003410' 254 00 0 00 003413'
 3348 003411' 260 17 0 00 013646'
 3349 003412' 254 00 0 00 004442'
 3350
 3351
 3352
 3353 003413' 201 03 0 00 001400
 3354 003414' 135 02 0 00 013655'
 3355 003415' 316 02 0 00 000003
 3356 003416' 254 00 0 00 003421'
 3357 003417' 260 17 0 00 013667'
 3358 003420' 254 00 0 00 004442'
 3359
 3360
 3361
 3362 003421' 201 03 0 00 177376
 3363 003422' 135 02 0 00 013711'
 3364 003423' 316 02 0 00 000003
 3365 003424' 254 00 0 00 003427'
 3366 003425' 260 17 0 00 013724'
 3367 003426' 254 00 0 00 004442'
 3368
 3369
 3370
 3371 003427' 135 02 0 00 013733'
 3372 003430' 316 02 0 00 000003
 3373 003431' 254 00 0 00 003434'
 3374 003432' 260 17 0 00 013746'
 3375 003433' 254 00 0 00 004442'
 3376
 3377
 3378
 3379 003434' 260 17 0 00 003370*

;Test response datagram opcode

```

      MOVEI   T1,^D69           ;Make correct 69
      MOVE    S1,RETBUF
      LDB     S2,[POINT 8,(S1),7] ;Put returned opcode in S2
      CAMN    S2,T1             ;Is returned status as expected ?
      JRST    T32AG2           ; Yes
      ERROR1  (3,T1,S2,Response datagram opcode in error,,NNN)
      JRST    TST32Y
  
```

;Test response datagram status

```

T32AG2: MOVEI   T1,0           ;Set up correct
      LDB     S2,[POINT 8,1(S1),15] ; S2=STATUS
      CAMN    S2,T1             ;Is returned status as expected ?
      JRST    T32AG3           ; Yes
      ERROR1  (3,T1,S2,Response datagram status in error,,NNN)
      JRST    TST32Y
  
```

;Test act count = 3.

```

T32AG3: MOVEI   T1,3*^D256     ;Set up correct
      LDB     S2,[POINT 16,1(S1),31] ; s2= act cnt.
      CAMN    S2,T1             ; test ACT CNT = EXPECTED.
      JRST    T32AG4           ; Yes
      ERROR1  (6,T1,S2,Response datagram account field in error,,NNN)
      JRST    TST32Y
  
```

;Test first two data bytes

```

T32AG4: MOVEI   T1,177376      ;Set up correct
      LDB     S2,[POINT 16,3(S1),31] ; s2=1ST 2 BYTES OF DATA.
      CAMN    S2,T1             ;Are first 2 bytes correct ?
      JRST    T32AG5           ; Yes
      ERROR1  (6,T1,S2,Response datagram first 2 data bytes in error,,NNN)
      JRST    TST32Y
  
```

;Test second two data bytes

```

T32AG5: LDB     S2,[POINT 16,4(S1),15] ; s2=2ND 2 BYTES OF DATA.
      CAMN    S2,T1             ;Are first 2 bytes correct ?
      JRST    T32AH           ; Yes
      ERROR1  (6,T1,S2,Response datagram second 2 data bytes in error,,NNN)
      JRST    TST32Y
  
```

;Requeue the buffer

T32AH: \$CALL REQUED

3380
 3381
 3382
 3383
 3384
 3385
 3386 003435' 201 01 0 00 000000
 3387 003436' 201 02 0 00 014046'
 3388 003437' 260 17 0 00 011372'
 3389
 3390
 3391
 3392 003440' 201 01 0 00 000005
 3393 003441' 202 01 0 00 003172*
 3394 003442' 202 01 0 00 003173*
 3395 003443' 201 01 0 00 000000
 3396 003444' 202 01 0 00 003176*
 3397 003445' 202 01 0 00 003200*
 3398 003446' 201 01 0 00 000003
 3399 003447' 202 01 0 00 003202*
 3400 003450' 201 01 0 00 000001
 3401 003451' 202 01 0 00 003175*
 3402 003452' 260 17 0 00 003203*
 3403
 3404
 3405
 3406 003453' 260 17 0 00 003204*
 3407
 3408
 3409
 3410 003454' 260 17 0 00 011242'
 3411 003455' 254 00 0 00 003460'
 3412 003456' 260 17 0 00 013547'
 3413 003457' 254 00 0 00 004442'
 3414
 3415 003460' 260 17 0 00 011317'
 3416 003461' 254 00 0 00 003464'
 3417 003462' 260 17 0 00 013571'
 3418 003463' 254 00 0 00 004442'
 3419
 3420 003464' 260 17 0 00 011341'
 3421 003465' 254 00 0 00 003470'
 3422 003466' 260 17 0 00 013610'
 3423 003467' 254 00 0 00 004442'
 3424
 3425
 3426
 3427 003470' 260 17 0 00 003375*

 ;* TST32B - DATAGRAM TEST. THIS TEST WILL SEND A DATAGRAM REQUEST OF
 ; DATA=BYTE PAIR, LENGTH=4, DELAY=0, REPEAT COUNT=3.
 ;*****

T32B: MOVEI S1,0 ;Load 0 for no message output
 MOVEI S2,[ASCIZ/Test 32 Part B - Datagram Test (Data=byte pairs, Repeat=3)/]
 \$CALL TRNODE ;Store that message please

;Build generate datagram ctp packet

MOVEI S1,5
 MOVEM S1,OPCODE ; opcode = gen DATAGRAM
 MOVEM S1,GENLEN ; gen length=4.
 MOVEI S1,0
 MOVEM S1,DELAY ; delay = 0
 MOVEM S1,GENCONST ; genconst=0.
 MOVEI S1,3
 MOVEM S1,RCOUNT ; repeat count = 3.
 MOVEI S1,1
 MOVEM S1,GENFUN ; genfunct =1/genfBPAIR
 \$CALL BLDCTP

;Send the datagram

\$CALL SNDDAT

;Wait for msg/dgm send complete and datagram available

\$CALL TEVPND ; Test for event pending.
 JRST T32BC1 ; No failure, continue
 ERROR1 (,,Timed out waiting for event pending after generate datagram sent,,NNN)
 JRST T32Y

T32BC1: \$CALL SNDCMP ; Test for msg/dgm send complete.
 JRST T32BC2 ; No failure, continue
 ERROR1 (,,Time out waiting for generate datagram send complete,,NNN)
 JRST T32Y

T32BC2: \$CALL DATAVL ; Wait for datagram available flag
 JRST T32BC3 ; No failure, continue
 ERROR1 (,,Time out waiting for generated datagram,,NNN)
 JRST T32Y

;Read the datagram

T32BC3: \$CALL READD

3428
 3429
 3430
 3431 003471' 201 03 0 00 000105
 3432 003472' 200 01 0 00 003377*
 3433 003473' 135 02 0 00 013617'
 3434 003474' 316 02 0 00 000003
 3435 003475' 254 00 0 00 003500'
 3436 003476' 260 17 0 00 013627'
 3437 003477' 254 00 0 00 004442'
 3438
 3439
 3440
 3441 003500' 201 03 0 00 000000
 3442 003501' 135 02 0 00 013636'
 3443 003502' 316 02 0 00 000003
 3444 003503' 254 00 0 00 003506'
 3445 003504' 260 17 0 00 013646'
 3446 003505' 254 00 0 00 004442'
 3447
 3448
 3449
 3450 003506' 201 03 0 00 000000
 3451 003507' 135 02 0 00 013655'
 3452 003510' 316 02 0 00 000003
 3453 003511' 254 00 0 00 003514'
 3454 003512' 260 17 0 00 013667'
 3455 003513' 254 00 0 00 004442'
 3456
 3457
 3458
 3459 003514' 201 03 0 00 000000
 3460 003515' 135 02 0 00 013711'
 3461 003516' 316 02 0 00 000003
 3462 003517' 254 00 0 00 003522'
 3463 003520' 260 17 0 00 013724'
 3464 003521' 254 00 0 00 004442'
 3465
 3466
 3467
 3468 003522' 201 03 0 00 000400
 3469 003523' 135 02 0 00 013733'
 3470 003524' 316 02 0 00 000003
 3471 003525' 254 00 0 00 003530'
 3472 003526' 260 17 0 00 013746'
 3473 003527' 254 00 0 00 004442'
 3474
 3475
 3476
 3477 003530' 260 17 0 00 003434*

;Test response datagram opcode

```

MOVEI T1,^D69 ;Make correct 69
MOVE S1,RETBUF
LDB S2,[POINT 8,(S1),7] ;Put returned opcode in S2
CAMN S2,T1 ;Is returned status as expected ?
JRST T32BD1 ; Yes
ERROR1 (3,T1,S2,Response datagram opcode in error,,NNN)
JRST TST32Y

```

;Test response datagram status

```

T32BD1: MOVEI T1,0 ;Set up correct
LDB S2,[POINT 8,1(S1),15] ; S2=STATUS
CAMN S2,T1 ;Is returned status as expected ?
JRST T32BD2 ; Yes
ERROR1 (3,T1,S2,Response datagram status in error,,NNN)
JRST TST32Y

```

;Test act count = 0.

```

T32BD2: MOVEI T1,0 ;Set up correct
LDB S2,[POINT 16,1(S1),31] ; s2= act cnt.
CAMN S2,T1 ; test ACT CNT = EXPECTED.
JRST T32BD3 ; Yes
ERROR1 (6,T1,S2,Response datagram account field in error,,NNN)
JRST TST32Y

```

;Test first two data bytes

```

T32BD3: MOVEI T1,0 ;Set up correct
LDB S2,[POINT 16,3(S1),31] ; s2=1ST 2 BYTES OF DATA.
CAMN S2,T1 ;Are first 2 bytes correct ?
JRST T32BD4 ; Yes
ERROR1 (6,T1,S2,Response datagram first 2 data bytes in error,,NNN)
JRST TST32Y

```

;Test second two data bytes

```

T32BD4: MOVEI T1,400 ;Set up correct
LDB S2,[POINT 16,4(S1),15] ; s2=2ND 2 BYTES OF DATA.
CAMN S2,T1 ;Are first 2 bytes correct ?
JRST T32BD5 ; Yes
ERROR1 (6,T1,S2,Response datagram second 2 data bytes in error,,NNN)
JRST TST32Y

```

;Requeue the buffer

T32BD5: \$CALL REQUED

3478
 3479
 3480
 3481 003531' 260 17 0 00 011220'
 3482 003532' 254 00 0 00 003535'
 3483 003533' 260 17 0 00 013610'
 3484 003534' 254 00 0 00 004442'
 3485
 3486
 3487
 3488 003535' 260 17 0 00 003470*
 3489
 3490
 3491
 3492 003536' 201 03 0 00 000105
 3493 003537' 200 01 0 00 003472*
 3494 003540' 135 02 0 00 013617'
 3495 003541' 316 02 0 00 000003
 3496 003542' 254 00 0 00 003545'
 3497 003543' 260 17 0 00 013627'
 3498 003544' 254 00 0 00 004442'
 3499
 3500
 3501
 3502 003545' 201 03 0 00 000000
 3503 003546' 135 02 0 00 013636'
 3504 003547' 316 02 0 00 000003
 3505 003550' 254 00 0 00 003553'
 3506 003551' 260 17 0 00 013646'
 3507 003552' 254 00 0 00 004442'
 3508
 3509
 3510
 3511 003553' 201 03 0 00 000400
 3512 003554' 135 02 0 00 013655'
 3513 003555' 316 02 0 00 000003
 3514 003556' 254 00 0 00 003561'
 3515 003557' 260 17 0 00 013667'
 3516 003560' 254 00 0 00 004442'
 3517
 3518
 3519
 3520 003561' 201 03 0 00 000000
 3521 003562' 135 02 0 00 013711'
 3522 003563' 316 02 0 00 000003
 3523 003564' 254 00 0 00 003567'
 3524 003565' 260 17 0 00 013724'
 3525 003566' 254 00 0 00 004442'

;Wait for datagram available

\$CALL WAITDA ; Wait for datagram available flag
 JRST T32BE1 ; No failure, continue
 ERROR1 (,Time out waiting for generated datagram,,NNN)
 JRST TST32Y

;Read the datagram

T32BE1: \$CALL READD

;Test response datagram opcode

MOVEI T1,^D69 ;Make correct 69
 MOVE S1,RETFUF
 LDB S2,[POINT 8,(S1),7] ;Put returned opcode in S2
 CAMN S2,T1 ;Is returned status as expected ?
 JRST T32BE2 ; Yes
 ERROR1 (3,T1,S2,Response datagram opcode in error,,NNN)
 JRST TST32Y

;Test response datagram status

T32BE2: MOVEI T1,0 ;Set up correct
 LDB S2,[POINT 8,1(S1),15] ; S2=STATUS
 CAMN S2,T1 ;Is returned status as expected ?
 JRST T32BE3 ; Yes
 ERROR1 (3,T1,S2,Response datagram status in error,,NNN)
 JRST TST32Y

;Test act count = 1.

T32BE3: MOVEI T1,1*^D256 ;Set up correct
 LDB S2,[POINT 16,1(S1),31] ; s2= act cnt.
 CAMN S2,T1 ; test ACT CNT = EXPECTED.
 JRST T32BE4 ; Yes
 ERROR1 (6,T1,S2,Response datagram account field in error,,NNN)
 JRST TST32Y

;Test first two data bytes

T32BE4: MOVEI T1,0 ;Set up correct
 LDB S2,[POINT 16,3(S1),31] ; s2=1ST 2 BYTES OF DATA.
 CAMN S2,T1 ;Are first 2 bytes correct ?
 JRST T32BE5 ; Yes
 ERROR1 (6,T1,S2,Response datagram first 2 data bytes in error,,NNN)
 JRST TST32Y

```

2006
2007
2008
2009
2010
2011
2012
2013
2014 002250' 304 00 0 00 000000 SSEVT:  $$SAVE    <S1,S2>                ;Save a couple ACs
2015
2016 002255' 201 02 0 00 002270'      MOVEI    S2,EVTARG                ;Arg block addr
2017 002256' 201 01 0 00 000010      MOVEI    S1,.LB EVT                ;Arg block length
2018 002257' 202 01 0 02 000000      MOVEM    S1,.SQLEN(S2)            ;Put in arg block
2019 002260' 202 03 0 02 000001      MOVEM    T1,.SQCID(S2)          ;    CID
2020 002261' 201 01 0 00 000025      MOVEI    S1,.SSEVT                ;Function code
2021
2022 002262' 332 00 0 00 002606'      SKIPE    DEBUG                ;FAKE IT?
2023 002263' 254 00 0 00 002266'      JRST     SSEVT1
2024
2025 002264' 260 17 0 00 002230*      $CALL    SCSS$                ;Do the JSYS
2026 002265' 254 00 0 00 002267'      JRST     SSEVTX                ; Error ret (evaluated in EVTSRV)
2027
2028 002266' 350 00 0 17 000000      SSEVT1: AOS      (P)                ;Ret+2
2029
2030 002267' 263 17 0 00 000000      SSEVTX: $RET
2031
2032 ; ARGUMENT BLOCK
2033
2034 002270' 000000 000010      EVTARG: 0,..LB EVT                ;Ret # wds processed,..block length
2035 002271' 000000 000000      0                ;CID or -1,  CID on return
2036 002272' 000000 000000      0                ;Reserved,..node#
2037 002273' 000000 000000      0                ;Returned event code
2038 002274' 000000 000000      0                ;Returned event data
2039 002275' 000000 000000      0                ;
2040 002276' 000000 000000      0                ;
2041 002277' 000000 000000      0                ;
2042 002300' 000000 000010      .LB EVT                ;Length of arg block

```

```

2043
2044
2045
2046
2047
2048
2049
2050
2051
2052 002301' 304 00 0 00 000000
2053
2054 002306' 201 02 0 00 002322'
2055 002307' 201 01 0 00 000003
2056 002310' 202 01 0 02 000000
2057 002311' 202 03 0 02 000001
2058 002312' 202 04 0 02 000002
2059 002313' 201 01 0 00 000003
2060
2061 002314' 332 00 0 00 002606'
2062 002315' 254 00 0 00 002320'
2063
2064 002316' 260 17 0 00 002264*
2065 002317' 254 00 0 00 002231*
2066
2067 002320' 350 00 0 17 000000
2068 002321' 263 17 0 00 000000
2069
2070
2071
2072 002322' 000000 000003
2073 002323' 000000 000000
2074 002324' 000000 000000
2075 002325' 000000 000003
2076

;*****
;* SSDIS - Disconnect from a remote node
; Enter with  T1 = CID (Connect ID)
;             T2 = Code of reason for disconnect
; Return+1    if jsys error
; Return+2    if successful
;*****

SSDIS:  $SAVE  <S1,S2>                ;Save a couple
        MOVEI  S2,DISARG                ;Arg block addr
        MOVEI  S1,..LBDIS              ;Arg block length
        MOVEM  S1,..SQLEN(S2)          ;Put in arg block
        MOVEM  T1,..SQCID(S2)          ; CID
        MOVEM  T2,..SQDIS(S2)          ; " reason for disconnect
        MOVEI  S1,..SSDIS              ;Function code

        SKIPE  DEBUG                    ;FAKE IT?
        JRST   SSDIS1

        $CALL  SCSS                      ;Do the JSYS
        JRST   SCSERR                    ;Print error message, return +1

SSDIS1: AOS      (P)                      ;Ret+2
        $RET

;ARGUMENT BLOCK

DISARG: 0,..LBDIS                        ;Ret #wds processed,,block length
        0                                ;CID
        0                                ;Disconnect code, reason for disconn
        .LBDIS                          ;Length of arg block

```



```

2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087 002326' 304 00 0 00 000000
2088
2089 002333' 201 02 0 00 002350'
2090
2091 002334' 201 01 0 00 000005
2092 002335' 202 01 0 02 000000
2093 002336' 200 01 0 00 000000*
2094 002337' 202 01 0 02 000004
2095 002340' 202 03 0 02 000001
2096 002341' 202 04 0 02 000002
2097 002342' 202 05 0 02 000003
2098 002343' 201 01 0 00 000006
2099
2100 002344' 260 17 0 00 002316*
2101 002345' 254 00 0 00 002317*
2102 002346' 350 00 0 17 000000
2103 002347' 263 17 0 00 000000
2104
2105
2106
2107 002350' 000000 000005
2108 002351' 000000 000000
2109 002352' 000000 000000
2110 002353' 000000 000000
2111 002354' 000000 000000
2112 002355' 000000 000005

;*****
; * SSSMG - Send a message to a remote node.
; Enter with:  T1 = CID (Connect ID)
;              T2 = Message buffer Address
;              T3 = Message buffer length in bytes
; Return+1    if jsys error
; Return+2    if successful
;*****

SSSMG:  $SAVE  <S1,S2>                ;Save two
        MOVEI  S2,SMGARG                ;Arg block addr
        MOVEI  S1,.LBSMG                ;Arg block length
        MOVEM  S1,.SQLEN(S2)            ;Put in arg block
        MOVE   S1,PATHS                  ;Load path to use
        MOVEM  S1,.SQFLG(S2)            ; and store in flag word
        MOVEM  T1,.SQCID(S2)            ; " CID
        MOVEM  T2,.SQAPT(S2)            ; " Addr of msg buffer
        MOVEM  T3,.SQLPT(S2)            ; " Length
        MOVEI  S1,.SSSMG                ;Function code

        $CALL  SCSS$                    ;Do the JSYS
        JRST   SCSERR                   ;Print error message, return +1
        AOS    (P)                      ;Ret+2
        $RET

; ARGUMENT BLOCK
SMGARG:  0,...LBSMG                    ;Ret #wds processed, block length
        0                                ;CID
        0                                ;Addr of msg text
        0                                ;Length of msg 8 bit bytes for ind std
        0                                ;Flags
        .LBSMG                          ;Length of arg block

```



```

2113
2114
2115      ;*****
2116      ;* SSQRM - Queue Message Buffers - receive a message from a DUP server.
2117      ;* Enter with  T1 = CID (Connect ID)
2118      ;*               T2 = Address of MSG buffer
2119      ;* Return+1    if jsys error
2120      ;* Return+2    if successful
2121      ;*****
2122 002356' 304 00 0 00 000000  SSQRM:  $SAVE  <S1,S2>          ;Save some ac's
2123
2124 002363' 201 02 0 00 002375'  MOVEI  S2,QRMARG          ;Arg block addr
2125 002364' 201 01 0 00 000003  MOVEI  S1,.LBQPM          ;Arg block length
2126 002365' 202 01 0 02 000000  MOVEM  S1,.SQLEN(S2)       ;Put in arg block
2127 002366' 202 03 0 02 000001  MOVEM  T1,.SQCID(S2)       ; " CID
2128 002367' 202 04 0 02 000002  MOVEM  T2,.SQAFB(S2)       ; " Addr of msg buffer
2129 002370' 201 01 0 00 000007  MOVEI  S1,.SSQRM          ;Function code
2130
2131 002371' 260 17 0 00 002344*   $CALL  SCSS$          ;Do the JSYS
2132 002372' 254 00 0 00 002345*   JRST  SCERR          ;Print error message, return +1
2133 002373' 350 00 0 17 000000  AOS     (P)          ;Ret+2
2134 002374' 263 17 0 00 000000  $RET
2135
2136      ; ARGUMENT BLOCK
2137
2138 002375' 000000 000003  QRMARG: 0,..LBQRM          ;Ret #wds processed,,block length
2139 002376' 000000 000000  0          ;CID
2140 002377' 000000 000000  0          ;Addr of 1st msg buff in msg buff chain
2141 002400' 000000 000003  .LBQRM          ;Length of arg block

```

```

2142
2143      ;#*****
2144      ;* SSCRM - Cancel (dequeue) Receive Message
2145
2146      ; Enter with  T1 = CID (Connect ID)
2147      ;              T2 = Address of Message Buffer
2148      ; Return+1    if jsys error
2149      ; Return+2    if successful
2150      ;#*****
2151
2152      002401' 304 00 0 00 000000      SSCRM:  $SAVE    <S1,S2>                ;Save some ac's
2153
2154      002406' 201 02 0 00 002420'      MOVEI    S2,CRMARG                ;Arg block addr
2155      002407' 201 01 0 00 000003      MOVEI    S1,..LBCRM                ;Arg block length
2156      002410' 202 01 0 02 000000      MOVEM    S1,..SQLEN(S2)            ;Put in arg block
2157      002411' 202 03 0 02 000001      MOVEM    T1,..SQCID(S2)            ; " CID
2158      002412' 202 04 0 02 000002      MOVEM    T2,..SQADB(S2)            ; " Addr of msg buffer
2159      002413' 201 01 0 00 000027      MOVEI    S1,..SSCRM                ;Function code
2160
2161      002414' 260 17 0 00 002371*      $CALL    SCSS$                ;Do the JSYS
2162      002415' 254 00 0 00 002372*      JRST     SCSEERR                ;Print error message, return +1
2163      002416' 350 00 0 17 000000      AOS      (P)                ;Ret+2
2164      002417' 263 17 0 00 000000      $RET
2165
2166      ; ARGUMENT BLOCK
2167
2168      002420' 000000 000003      CRMARG:  0,..LBCRM                ;Ret #wds processed,,block length
2169      002421' 000000 000000      0                ;CID
2170      002422' 000000 000000      0                ;Addr of buffer to dequeue
2171      002423' 000000 000003      .LBCRM                ;Length of arg block
  
```

```

2172
2173
2174
2175
2176
2177
2178
2179
2180
2181 002424' 304 00 0 00 000000
2182
2183 002431' 201 02 0 00 002443'
2184 002432' 201 01 0 00 000007
2185 002433' 202 01 0 02 000000
2186 002434' 202 03 0 02 000001
2187 002435' 202 04 0 02 000004
2188 002436' 201 01 0 00 000010
2189
2190 002437' 260 17 0 00 002414*
2191 002440' 254 00 0 00 002415*
2192 002441' 350 00 0 17 000000
2193 002442' 263 17 0 00 000000
2194
2195
2196
2197 002443' 000000 000007
2198 002444' 000000 000000
2199 002445' 000000 000000
2200 002446' 000000 000000
2201 002447' 000000 000000
2202 002450' 000000 000000
2203 002451' 000000 000000
2204 002452' 000000 000007

;*****
;* SSCSP - Connect State Poll - status of connection
; Enter with T1 = CID (Connect ID)
;            T2 = Byte pointer to dest process name
; Return+1   if jsys error
; Return+2   if successful
;*****

SSCSP: $SAVE <S1,S2> ;Save some ac's

        MOVEI S2,CSPARG ;Arg block addr
        MOVEI S1,.LBCSP ;Arg block length
        MOVEM S1,.SQLEN(S2) ;Put in arg block
        MOVEM T1,.SQCID(S2) ;CID
        MOVEM T2,.SQBDN(S2) ;Dest name byte pointer
        MOVEI S1,.SSCSP ;Function code

        $CALL SCSS$ ;Do the JSYS
        JRST SCSERR ;Print error message, return +1
        AOS (P) ;Ret+2
        $RET

; ARGUMENT BLOCK

CSPARG: 0,..LBCSP ;Ret #wds processed,..block length
        0 ;CID
        0 ;Ret connection state
        0 ;Ret dest connect
        0 ;Ret byte ptr of loc to start dest name
        0 ;Ret node#
        0 ;Ret src disconn code,..dst disconn code
        .LBCSP ;Length of arg block

```

```

2205
2206 ;*****
2207 ;* SSSTS - Status Info on Connection (Fast form of .SSCSP)
2208 ; Enter with T1 = CID (Connect ID)
2209 ; Return+1 if jsys error
2210 ; Return+2 if successful
2211 ;*****
2212
2213 002453' 304 00 0 00 000000 SSSTS: $SAVE <S1,S2> ;Save some ac's
2214
2215 002460' 201 02 0 00 002473' MOVEI S2,STSARG ;Arg block addr
2216 002461' 201 01 0 00 000004 MOVEI S1,.LBSTS ;Arg block length
2217 002462' 202 01 0 02 000000 MOVEM S1,.SQLEN(S2) ;Put in arg block
2218 002463' 202 03 0 02 000001 MOVEM T1,.SQCID(S2) ; CID
2219 002464' 201 01 0 00 000012 MOVEI S1,.SSSTS ;Function code
2220
2221 002465' 332 00 0 00 002606' SKIPE DEBUG ; FAKE IT?
2222 002466' 254 00 0 00 002471' JRST SSSTS1 ; YES
2223
2224 002467' 260 17 0 00 002437* $CALL SCSS$ ;Do the JSYS
2225 002470' 254 00 0 00 002472' JRST SSSTSX ; Error ret (evaluated in CALLING RTN)
2226
2227 002471' 350 00 0 17 000000 SSSTS1: AOS (P) ;Ret+2
2228
2229 002472' 263 17 0 00 000000 SSSTSX: $RET
2230
2231 ; Argument block
2232
2233 002473' 000000 000004 STSARG: 0,...LBSTS ;Ret #wds processed,.block length
2234 002474' 000000 000000 0 ;CID
2235 002475' 000000 000000 0 ;Ret flags,.connect state code
2236 002476' 000000 000000 0 ;reserved,.node#
2237 002477' 000000 000004 .LBSTS ;Length of arg block

```

```

2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248 002500' 304 00 0 00 000000
2249
2250 002505' 201 02 0 00 002516'
2251 002506' 201 01 0 00 000005
2252 002507' 202 01 0 02 000000
2253 002510' 202 03 0 02 000001
2254 002511' 201 01 0 00 000013
2255
2256 002512' 260 17 0 00 002467*
2257 002513' 254 00 0 00 002440*
2258 002514' 350 00 0 17 000000
2259 002515' 263 17 0 00 000000
2260
2261
2262
2263 002516' 000000 000005
2264 002517' 000000 000000
2265 002520' 000000 000000
2266 002521' 000000 000000
2267 002522' 000000 000000
2268 002523' 000000 000005

;*****
;* SSRMG - Receive Message
;* Data received is placed in a user's buffer space previously queued.
;* SCS returns the address of the buffer in .SQARB of the jsys arg. block
;* Enter with T1 = CID (Connect ID)
;* Return+1 if jsys error
;* Return+2 if successful
;*****

SSRMG: $SAVE <S1,S2> ;Save some ac's

        MOVEI S2,RMGARG ;Arg block addr
        MOVEI S1,.LBRMG ;Arg block length
        MOVEM S1,.SQLEN(S2) ;Put in arg block
        MOVEM T1,.SQCID(S2) ;CID
        MOVEI S1,.SSRMG ;Function code

        $CALL SCSS ;Do the JSYS
        JRST SCSEERR ;Print error message, return +1
        AOS (P) ;Ret+2
        $RET

; ARGUMENT BLOCK

RMGARG: 0,.LBRMG ;Ret #wds processed,.block length
        0 ;CID or -1
        0 ;Ret addr of msg buffer
        0 ;Ret flags,.node#
        0 ;Ret len of msg
        .LBRMG ;Length of arg block

```



```

2269
2270
2271
2272
2273
2274
2275
2276
2277 002524' 304 00 0 00 000000
2278
2279 002531' 201 02 0 00 002542'
2280 002532' 201 01 0 00 000003
2281 002533' 202 01 0 02 000000
2282 002534' 202 03 0 02 000001
2283 002535' 201 01 0 00 000024
2284
2285 002536' 260 17 0 00 002512*
2286 002537' 254 00 0 00 002513*
2287 002540' 350 00 0 17 000000
2288 002541' 263 17 0 00 000000
2289
2290
2291
2292 002542' 000000 000003
2293 002543' 000000 000000
2294 002544' 000000 000000
2295 002545' 000000 000003

;*****
;* SSGDE - Get Entry from Data Queue
; Enter with T1 = CID (Connect ID)
; Return+1 if jsys error
; Return+2 if successful
;*****
SSGDE: $SAVE <S1,S2> ;Save some ac's
        MOVEI S2,GDEARG ;Arg block addr
        MOVEI S1,.LBGDE ;Arg block length
        MOVEM S1,.SQLEN(S2) ;Put in arg block
        MOVEM T1,.SQCID(S2) ; CID
        MOVEI S1,.SSGDE ;Function code
        $CALL SCS$ ;Do the JSYS
        JRST SCSERR ;Print error message, return +1
        ADS (P) ;Ret+2
        $RET

;ARGUMENT BLOCK
GDEARG: 0,..LBGDE ;Ret #wds processed,..block length
        0 ;CID or -1
        0 ;Ret buff ID which completed DMA xfer
        .LBGDE ;Length of arg block
  
```

2296
 2297
 2298
 2299
 2300
 2301
 2302
 2303
 2304 002546' 304 00 0 00 000000
 2305 002553' 402 00 0 00 002601'
 2306 002554' 201 01 0 00 000005
 2307 002555' 202 01 0 00 002577'
 2308 002556' 200 01 0 00 002576'
 2309 002557' 202 01 0 00 002602'
 2310
 2311
 2312
 2313 002560' 201 01 0 00 000004
 2314 002561' 332 00 0 00 000125*
 2315 002562' 435 01 0 00 000010
 2316 002563' 202 01 0 00 002600'
 2317
 2318 002564' 201 01 0 00 000014
 2319 002565' 201 02 0 00 002577'
 2320 002566' 260 17 0 00 002536*
 2321 002567' 254 00 0 00 002537*
 2322 002570' 200 01 0 00 002216*
 2323 002571' 260 17 0 00 000704'
 2324 002572' 200 02 0 00 002601'
 2325 002573' 202 02 0 01 000005
 2326 002574' 351 00 0 17 000000
 2327 002575' 263 17 0 00 000000
 2328
 2329 002576' 000000 000000
 2330
 2331 002577' 000000 000005
 2332 002600' 000000 000000
 2333 002601' 000000 000000
 2334 002602' 000000 000000
 2335 002603' 000000 000000

```

*****
;Map a buffer
; Call via '$CALL SSMAP'. This subroutine maps a buffer AS SPECIFIED IN
; MAPBLN(MAP BUFFER LENGTH) . The returned buffer name is stored in
; the fifth word of the DEVTBL.
*****
SSMAP: $SAVE <S1,S2> ;Save ACs
      SETZM MAPBF+.SQBNA ;Zero returned buffer name
      MOVEI S1,5 ;Load length
      MOVEM S1,MAPBF+.SQLEN ; and save it here
      MOVE S1,MAPBLN ;Get length of buffer to map
      MOVEM S1,MAPBF+.SQBNA+1 ;Put it in the argument block

;SET 'NOT CLEAR BIT IN FLAG WORD'

      MOVEI S1,SQ%WRT ;Load write enable bit
      SKIPE FLGFLG ;Set do-not-clear-valid?
      ORI S1,SQ%CVD ;Yes
      MOVEM S1,MAPBF+.SQXFL ;Save flags word

SSMAP1: MOVEI S1,SSMAP ;Function code
      MOVEI S2,MAPBF ;Argument block address
      $CALL SCSS ;Do the JSYS
      JRST SCSERR ;Give +1 return
      MOVE S1,NODEN ;Get node number we are working on
      $CALL DTBLE ;Point to table entry in S1
      MOVE S2,MAPBF+.SQBNA ;Get returned buffer name
      MOVEM S2,DEVIMB(S1) ;Save it in DEVTBL (5th entry)
      AOS (P)
      $RET ;Return to calling

MAPBLN: 0 ;Supplied by CITSTP.MAC

MAPBF: 0,,5 ;Processed words,,length of block
      0 ;Flags
      0 ;Returned buffer name
      0 ;Buffer length in bytes
      0 ;Address of buffer segment
  
```

```

2336                                SUBTTL  Exerciser Storage
2337
2338 002604' 000000 000000      TPSFLG: 0                ; -1 if TOPS-20/TOPS-10 responder
2339 002605' 000000 000125      MAPWRD: 125            ; Byte to fill buffers with
2340 002606' 000000 000000      DEBUG: 0              ; Debug set to avoid JS's execution
2341 002607' 000000 000000      DGMBFL: 0
2342 002610' 000000 000000      MSGBFL: 0
2343 002611' 000000 000000      DGMNXT: 0
2344 002612' 000000 000000      MSGNXT: 0
2345 002613' 000000 000000      NDSHFT: 0
2346 002614' 000000 000000      CONSTA: 0              ; Connection status
2347 002615' 000000 000000      EVTCOD: 0
2348 002616' 000000 000000      EVTNOD: 0
2349 002617' 000000 000000      EVTCID: 0
2350 002620' 000000 000000      READND: 0
2351 002621' 000000 000000      TCOUNT: 0
2352 002622' 000000 000000      TLENG: 0              ; Bytes of data for this test
2353 002623' 000000 000000      NODEBM: 0              ; Bitmask of node under test
2354 002624' 000000 000000      TNODED: 0
2355 002625' 000000 000000      LSTBFR: 0
2356 002626' 000000 000000      FSTBFR: 0
2357 002627' 000000 000000      NUMBFR: 0
2358 002630' 000000 000000      CMPRWD: 0
2359 002631' 000000 000000      CPYCNT: 0
2360 002632' 000000 000000      NODADR: 0
2361 002633' 000000 000000      CNTNOD: 0
2362 002634' 000000 000000      LINKBF: 0
2363 002635' 000000 000000      ERRFLG: 0
2364 002636' 000000 000000      MPXERR: 0
2365 002637' 000000 000000      MVBERR: 0
2366 002640' 000000 000000      UMPERR: 0
2367 002641' 000000 000000      RDMERR: 0
2368 002642' 000000 000000      RDDERR: 0
2369 002643' 000000 000000      DGMERR: 0
2370 002644' 000000 000000      MSGERR: 0
2371 002645' 000000 000000      DMAPTR: BLOCK 1        ; Pointer to DMA buffers from MAPBBA
  
```

2372
 2373
 2374
 2375
 2376
 2377
 2378
 2379
 2380
 2381
 2382
 2383
 2384
 2385
 2386
 2387
 2388
 2389

002646' 000000 000000

002647' 000000 000000
 002650'

;NODTBL HAS EACH BIT (STARTING AT BIT 0) SET TO INDICATE THAT
 ;THAT PARTICULAR NODE EXISTS. THE BITS ARE SET IN RESPONSE TO
 ;THE RCD COMMAND BEING SUCCESSFUL.

:	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	16-35
:	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	

NODTBL: 0 ;Bitmask of available nodes
 ;TSTNOD HAS EACH BIT (STARTING AT BIT 0) SET TO INDICATE THAT
 ;THAT PARTICULAR NODE IS TO BE INCLUDED IN THE PERFORMANCE TEST.
 ;IT IS THE 'LOGICAL AND OF THE 'NODTBL' BIT AND THE UNIT SELECTED BIT.
 TSTNOD: 0 ;Bitmask of selected nodes
 TNODNU: BLOCK 1 ;Number of nodes selected

2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436

;DEVICE TABLE CONTENTS:

:	+	-----	+
:		CID	:
:	+	-----	+
:		POINTER TO CONNECTION DATA	:
:	+	-----	+
:		Message Counter	:
:	+	-----	+
:		POINTER TO SAVED CTP PACKET	:
:	+	-----	+
:		RETURNED BUFFER NAME FROM MAP BFR CMD	:
:	+	-----	+
:		RETURNED BUFFER NAME FROM INTERNALLY MAPPED BUFFER	:
:	+	-----	+
:		ADRS OF INTERNALLY MAPPED BUFFER	:
:	+	-----	+

000000
000002
000003
000004
000005
000006
000007

DEVCID==0
DEVCON==1
DEVCNT==2
DEVPKT==3
DEVBNM==4
DEVIMB==5
DEVIMA==6
DTBLNG==7

;Connect ID
;Pointer to connect data
;Iteration/message counter
;Saved packet
;Externally mapped buffer name
;Internally mapped buffer name
;Internally mapped buffer addr
;DEVTBL length

DEVTBL:
DEVT00: BLOCK DTBLNG
DEVT01: BLOCK DTBLNG
DEVT02: BLOCK DTBLNG
DEVT03: BLOCK DTBLNG
DEVT04: BLOCK DTBLNG
DEVT05: BLOCK DTBLNG
DEVT06: BLOCK DTBLNG
DEVT07: BLOCK DTBLNG
DEVT08: BLOCK DTBLNG
DEVT09: BLOCK DTBLNG
DEVT10: BLOCK DTBLNG
DEVT11: BLOCK DTBLNG
DEVT12: BLOCK DTBLNG
DEVT13: BLOCK DTBLNG
DEVT14: BLOCK DTBLNG
DEVT15: BLOCK DTBLNG
DEVEND==.-1

;Node 0 storage
;Node 1 storage
;Node 2 storage
;Node 3 storage
;Node 4 storage
;Node 5 storage
;Node 6 storage
;Node 7 storage
;Node 8 storage
;Node 9 storage
;Node 10 storage
;Node 11 storage
;Node 12 storage
;Node 13 storage
;Node 14 storage
;Node 15 storage
;Last location to clear

002651'
002651'
002660'
002667'
002676'
002705'
002714'
002723'
002732'
002741'
002750'
002757'
002766'
002775'
003004'
003013'
003022'

003030'

2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455

003031'
003035'
003041'
003045'
003051'
003055'
003061'
003065'
003071'
003075'
003101'
003105'
003111'
003115'
003121'
003125'

;SAVE AREA FOR CTP MESSAGE REQUEST PACKETS.

MSGSV0: BLOCK 4
MSGSV1: BLOCK 4
MSGSV2: BLOCK 4
MSGSV3: BLOCK 4
MSGSV4: BLOCK 4
MSGSV5: BLOCK 4
MSGSV6: BLOCK 4
MSGSV7: BLOCK 4
MSGSV8: BLOCK 4
MSGSV9: BLOCK 4
MSGSVA: BLOCK 4
MSGSVB: BLOCK 4
MSG SVC: BLOCK 4
MSGSVD: BLOCK 4
MSGSVE: BLOCK 4
MSGSVF: BLOCK 4

; UNIT 00 MESSAGE SAVE AREA.
; UNIT 01 MESSAGE SAVE AREA.
; UNIT 02 MESSAGE SAVE AREA.
; UNIT 03 MESSAGE SAVE AREA.
; UNIT 04 MESSAGE SAVE AREA.
; UNIT 05 MESSAGE SAVE AREA.
; UNIT 06 MESSAGE SAVE AREA.
; UNIT 07 MESSAGE SAVE AREA.
; UNIT 08 MESSAGE SAVE AREA.
; UNIT 09 MESSAGE SAVE AREA.
; UNIT 10 MESSAGE SAVE AREA.
; UNIT 11 MESSAGE SAVE AREA.
; UNIT 12 MESSAGE SAVE AREA.
; UNIT 13 MESSAGE SAVE AREA.
; UNIT 14 MESSAGE SAVE AREA.
; UNIT 15 MESSAGE SAVE AREA.

2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475

003131'
003155'
003201'
003225'
003251'
003275'
003321'
003345'
003371'
003415'
003441'
003465'
003511'
003535'
003561'
003605'

000024

;RCD Data Blocks

RCDLEN==.LBRCD
RCV00: BLOCK RCDLEN
RCV01: BLOCK RCDLEN
RCV02: BLOCK RCDLEN
RCV03: BLOCK RCDLEN
RCV04: BLOCK RCDLEN
RCV05: BLOCK RCDLEN
RCV06: BLOCK RCDLEN
RCV07: BLOCK RCDLEN
RCV10: BLOCK RCDLEN
RCV11: BLOCK RCDLEN
RCV12: BLOCK RCDLEN
RCV13: BLOCK RCDLEN
RCV14: BLOCK RCDLEN
RCV15: BLOCK RCDLEN
RCV16: BLOCK RCDLEN
RCV17: BLOCK RCDLEN

: UNIT 0 RCD DATA BLOCK.
: UNIT 1 RCD DATA BLOCK.
: UNIT 2 RCD DATA BLOCK.
: UNIT 3 RCD DATA BLOCK.
: UNIT 4 RCD DATA BLOCK.
: UNIT 5 RCD DATA BLOCK.
: UNIT 6 RCD DATA BLOCK.
: UNIT 7 RCD DATA BLOCK.
: UNIT 8 RCD DATA BLOCK.
: UNIT 9 RCD DATA BLOCK.
: UNIT 10 RCD DATA BLOCK.
: UNIT 11 RCD DATA BLOCK.
: UNIT 12 RCD DATA BLOCK.
: UNIT 13 RCD DATA BLOCK.
: UNIT 14 RCD DATA BLOCK.
: UNIT 15 RCD DATA BLOCK.

2476					
2477		000100		MAXBYT==100	:64. bytes max in a buffer
2478		000021		BUFSIZ==1+<MAXBYT/4>	:Size of each buffer below
2479					
2480				;MSGPTR is the message pointer. It points to the latest available	
2481				;buffer for messages. MSG000 is the first message buffer.	
2482					
2483	003631'	000000	003632'	MSGPTR: MSG000	;Pointer to buffers below to alloc
2484	003632'			MSG000: BLOCK BUFSIZ*^D15*^D32	;Buffer size*max other nodes*max msgs
2485		023571'		MSGEND==.-1	;Last loc to clear
2486					
2487				;Datagram buffers. DGMPiR points to next free buffer, DGM000 is first one.	
2488					
2489	023572'	000000	023573'	DGMPTR: DGM000	
2490	023573'			DGM000: BLOCK BUFSIZ*^D15*^D32	;Buffer size*max other nodes*max dgms
2491		043532'		DGMEND==.-1	;Last loc to clear

DFCIBY Test 99 - Performance Exerciser version 2(20) MACRO %53B(1242) 17:41 27-Aug-85 Page 72
DFCIBY MAC 22-Aug-85 23:07 Literals

SEQ 0641

2492
2493
2494
2495
2496
2497

043533'

SUBTTL Literals

LIT..Y: XLIST
LIST

;LIT

END

NO ERRORS DETECTED

PROGRAM BREAK IS 045714
CPU TIME USED 00:35.127

160P CORE USED

[illegible]

GENFUN	137#	809	1250											
GENLEN	137#	811												
GETCID	445	506	522	565	667	734	877#	1164	1332	1582	1757			
GETNOD	410	443	504	520	563	665	732	847	917#	1157	1328	1574	1755	
GEVNT	131#													
GTNOD1	918	921#												
GTNODX	919	928#												
HSCDST	150#	1363												
INITEX	233	303#												
INTUM1	1627	1628#												
INTUM2	1630#	1635												
ISDGER	670	680#												
ISDGM1	692	701#												
ISDMA1	1327#	1341	1349											
ISDMA2	1329	1343#												
ISDMAX	1346#													
ISMSG1	541	548#												
ISMVER	1336	1338	1340	1348#										
ISSDG1	663#	672	681											
ISSDG2	666	674#												
ISSDGM	669	685#												
ISSDGX	676	678#												
ISSDMA	284	1324#												
ISSMAP	278	1153#												
ISSMG1	519#	527												
ISSMG2	521	529#												
ISSMGX	531#													
ISSMP1	1156#	1172												
ISSMP2	1158	1176#												
ISSMP3	1177	1182#												
ISSMP4	1178	1183#												
ISSMPX	1168	1184#												
ISSMSG	525	536#												
ISSUED	262	659#												
ISSUEM	248	516#												
ISSUMP	291	1570#												
ISUMP1	1573#	1591												
ISUMP2	1575	1595#												
ISUMP3	1596	1601#												
ISUMP4	1597	1607#												
ISUMPX	1603	1609#												
ITCNT	143#	162	206	213										
LINKBF	316	1808	1823	2362#										
LIT..Y	2494#													
LNODE	132#	1393	1486											
LNODEN	132#	1405	1500											
LOFSET	139#	1401	1495											
LSTBFR	1810	1825	1847	1859	1878	1890	2355#							
LSTIN.	119	168	205	218	227	228	398	459	886	886#	900	900#	936	943
	944	993	993#	10										

[illegible]

[illegible]

RCV15	2473#									
RCV16	2474#									
RCV17	2475#									
RCVACT	150#	611	771	1289	1454	1549	1688	1772		
RCVDM	286	1718#	1779							
RCVMER	604	608	612	616	632#	1765	1769	1773	1777	
RCVRM	149#									
RCVRSP	148#	603	764	1092	1281	1446	1541	1680	1764	
RCVSTA	149#	607	767	1099	1285	1450	1545	1684	1768	
RDDERR	793	2368#								
RDDGM1	731#	742	748	753						
RDDGM2	738	744#								
RDDGM3	733	750#								
RDDGM4	752	755#								
RDDGMS	265	727#								
RDDMA1	1754#	1780	1785							
RDDMA2	1761	1779#								
RDDMA3	1756	1782#								
RDDMAS	1750#									
RDDMAX	1784	1787#								
RDDTO1	741	758#								
RDMERR	579	632	586							
RDMSG1	562#	573								
RDMSG2	569	575#								
RDMSG3	564	583#								
RDMSG4	585	588#								
RDMSG5	251	558#								
RDMT02	1436	1479#	1531							
RDMT03	1271	1317#								
RDMT04	1669	1711#								
RDSTIM	131#	132#	560	729	994	1266	1431	1525	1664	1752
READD	138#	1082								
READM	137#	1071								
READND	2350#									
RECDER	765	768	775	792#	799					
RECDG1	762	777#								
RECDG2	781#	787								
RECEVD	747	761#								
RECEVM	578	599#								
RECV1	600	618#								
RECV2	621#	627								
RECV3	629#	634	637							
REQUE	149#	789	794							
REQUEM	138#	552	629	705	1301	1314	1461	1476	1556	1707
RETBUF	137#	619	778	1088	1126					
RMGARG	2250	2263#								
RMOVX	1339	1393#								
RMOVX1	1422	1431#								
RMOVX2	1432#	1437								
RMOVX3	1433	1439#								
RMOVX4	1443	1461#	1470	1482						
RMOVXX	1463	1465#								
RMSGER	572	580	593#	758						

RMVXER	1447	1451	1455	1459	1469#	1553	1692	1776						
RNVNUM	149#	615	774	1293	1458									
ROFSET	139#	1402	1496											
RRBIN1	503#	511												
RRBIND	299	500#												
RRBINS	238	406#												
RSCSE	131#	345	372											
RSTDB1	1883#	1891												
RSTMB1	1852#	1860												
RSTRDB	1824	1869#												
RSTRMB	1809	1838#												
RTCHK	144#	234	1000	1017										
S1	155	157	162	163	177	178	184	185	192	193	194	195	196	205
	206	213	214	245	246	259	260	275	276	305	306	307	308	309
	310	314	315	316	317	318	323	324	329	330	335	336	361	363
	380	381	383	389	401	406	407	417	420	446	483	484	491	492
	500	501	507	523	536	537	558	559	566	583	594	596	619	620
	659	660	668	685	686	727	728	735	750	778	779	804	805	806
	807	808	809	810	811	817	822	831	833	842	850	851	860	861
	862	870	871	877	878	885	886	888	894	921	922	932	933	950
	955	956	964	969	970	978	983	984	992	995	996	1007	1015	1023
	1024	1046	1047	1058	1085	1086	1087	1088	1089	1096	1126	1127	1129	1133
	1135	1165	1191	1192	1194	1195	1202	1208	1214	1243	1244	1245	1246	1247
	1248	1249	1250	1251	1252	1274	1275	1296	1299	1333	1357	1358	1359	1363
	1374	1375	1394	1395	1396	1397	1398	1399	1400	1401	1402	1403	1404	1405
	1406	1410	1411	1412	1414	1416	1488	1489	1491	1492	1493	1494	1495	1496
	1498	1499	1500	1501	1505	1506	1507	1509	1511	1581	1583	1620	1622	1648
	1650	1720	1721	1724	1725	1729	1731	1732	1733	1734	1758	1782	1795	1808
	1823	1839	1840	1851	1852	1855	1870	1871	1882	1883	1886	1902	1904	1905
	1908	1919	1920	1921	1961	1962	1986	1987	2014	2017	2018	2020	2052	2055
	2056	2059	2087	2091	2092	2093	2094	2098	2122	2125	2126	2129	2152	2155
	2156	2159	2181	2184	2185	2188	2213	2216	2217	2219	2248	2251	2252	2254
	2277	2280	2281	2283	2304	2306	2307	2308	2309	2313	2315	2316	2318	2322
	2325													
S2	156	158	215	225	226	227	228	394	395	396	409	419	420	424
	425	426	427	442	503	519	562	620	622	664	731	779	782	819
	820	821	822	823	824	825	830	831	834	843	844	846	851	852
	854	885	888	890	892	899	900	902	904	917	926	954	955	956
	968	969	970	982	983	984	992	1004	1005	1006	1007	1021	1022	1023
	1024	1046	1051	1052	1053	1054	1055	1056	1057	1058	1089	1090	1092	1096
	1097	1099	1127	1128	1130	1131	1135	1136	1138	1153	1154	1156	1207	1208
	1209	1213	1214	1219	1223	1298	1299	1324	1325	1327	1414	1415	1416	1417
	1509	1510	1511	1512	1570	1571	1573	1622	1623	1639	1640	1642	1643	1645
	1646	1650	1651	1672	1673	1674	1675	1731	1732	1734	1737	1750	1751	1754
	1795	1799	1800	1802	1807	1814	1815	1817	1822	1838	1840	1844	1845	1846
	1847	1849	1850	1852	1854	1859	1869	1871	1875	1876	1877	1878	1880	1881
	1883	1885	1890	1902	1903	1905	1906	1907	1961	1963	1966	1968	1970	1972
	1974	1976	1978	2014	2016	2018	2019	2052	2054	2056	2057	2058	2087	2089
	2092	2094	2095	2096	2097	2122	2124	2126	2127	2128	2152	2154	2156	2157
	2158	2181	2183	2185	2186	2187	2213	2215	2217	2218	2248	2250	2252	2253
	2277	2279	2281	2282	2304	2319	2324	2325						
SC%DGA	964													
SC%DTA	978													

SC%EVA	483													
SC%MSA	950													
SCSS	133#	1913	1983	2025	2064	2100	2131	2161	2190	2224	2256	2285	2320	
SCSERC	133#	1919												
SCSERR	132#	953	967	981	1003	1020	1923	1984	2065	2101	2132	2162	2191	2257
	2286	2321												
SCSERS	132#	1190	1619	1625										
SCSISB	1921													
SELND1	409#	428	433											
SELND3	415#													
SELND4	416	432#												
SELNOD	415	1795#												
SETCNT	899#	1439	1534											
SMGARG	2089	2107#												
SNDCMP	149#	648	716											
SNDDAT	138#	708												
SNDMSG	138#	642												
SNODET	143#	389												
SQ%CVD	2315													
SQ%OPN	472	474												
SQ%WRT	2313													
SSCON	1827	1961#												
SSCON1	1981	1986#												
SSCRM	2152#													
SSCSP	2181#													
SSDIS	509	2052#												
SSDIS1	2062	2067#												
SSEVT	1049	2014#												
SSEVT1	2023	2028#												
SSEVTX	2026	2030#												
SSGDE	2277#													
SSINN%	1920													
SSMAP	1197	2304#												
SSMAP1	2318#													
SSQRM	2122#													
SSRCD	346	1902#												
SSRCD1	1914	1919#												
SSRCD3	1911	1916#												
SSRMG	2248#													
SSSMG	2087#													
SSSTS	952	966	980	1002	1019	2213#								
SSSTS1	2222	2227#												
SSSTSX	2225	2229#												
STSARG	954	968	982	1004	1021	2215	2233#							
SWTRAC	144#	166	211											
T%TEXT	167	204	217	397	458	935	942	943	1034	1039				
T1	343	384	391	395	400	507	618	621	626	777	781	786	824	825
	854	855	885	889	891	892	893	899	901	903	904	905	923	924
	925	926	951	965	979	1001	1018	1048	1129	1130	1138	1139	1218	1224
	1225	1727	1736	1740	1795	1801	1802	1803	1816	1817	1818	1902	1906	1951
	1965	1966	1967	1968	1969	1970	1971	1972	1973	1974	1975	1975	1977	1978
	2019	2057	2095	2127	2157	2186	2218	2253	2282					
T10DST	150#	1375												

	343	352	358	367	368	508	852	853	855	856	885	921	924	1902	SEQ 0650
T2	1907	1961	2058	2096	2128	2158	2187								
T20DST	150#	1374													
T3	352	353	409	440	470	471	472	474	516	517	622	623	782	783	
	798	850	856	1136	1137	1139	1141	1220	1221	1223	1228	1229	1230	1737	
	1738	1747	2097												
T4	351	353	354	358	359	360	361	362	363	885	899	992	1046	1795	
	1840	1871	1902	1961	2014	2052	2087	2122	2152	2181	2213	2248	2277	2304	
T99A	179	186	197	233#											
T99D	241	249	252	257	263	266	273	279	280	285	287	292	293	298#	
T99PSC	165	216	225#												
TCNT	584	751	842#	1783											
TCNT1	846#	857													
TCNT2	860#														
TCNTEX	848	865#													
TCOUNT	178	185	196	314	806	900	2351#								
TEVPND	148#	643	712												
TF	917	918													
TITCNT	143#	205													
TLENG	246	260	276	626	786	800	1191	1251	1403	1498	1645	2352#			
TNODED	407	409	440	442	501	503	517	519	559	562	583	660	664	728	
	731	750	1154	1156	1325	1327	1571	1573	1751	1754	1782	2354#			
TNODNU	192	379	399	2389#											
TNODT1	389#	401													
TNODT2	390	394#													
TNODT3	396	399#													
TNODT4	392	400#													
TNODTB	173	379#													
TPSFLG	1368	1381	1718	2338#											
TST99	127	155#													
TST99A	163	177#													
TST99B	203	211#													
TST99X	212	215	221#												
TST99Y	180	187	198	202#											
TSTACT	149#	1105													
TSTDEV	1335	1357#													
TSTDSC	143#	158													
TSTDV1	1364	1374#													
TSTDV2	1377	1387#													
TSTDVX	1372	1383	1389#												
TSTNAM	143#	157													
TSTNOD	202	240	256	272	381	391	406	439	500	516	558	659	727	843</	

UMPEXT	1588	1639#					
UMPEXX	1696	1698#	1704	1709	1714		
UMPINT	1585	1619#					
UMPXER	1681	1685	1689	1693	1703#		
VEDIT	114						
VERFCN	239	439#					
VRFCN1	442#	448					
WAITC	485	992#					
WAITC1	1000#	1011					
WAITC2	996	1015#					
WAITC3	1017#	1028					
WAITC8	1008	1025	1030#				
WAITE	53	1046#					
WCERR3	1010	1033#					
WCERR4	1027	1037#					
XMITER	542	545	551#				
XMOVR	1337	1486#					
XMOVR1	1517	1525#					
XMOVR2	1527#	1532					
XMOVR3	1528	1534#					
XMOVR4	1538	1556#	1565				
XMOVRX	1558	1560#					
XMTDER	694	698	704#				
XMTDRA	713	716#					
XMTDRB	715	717	721	722#			
XMTDRQ	690	708#					
XMTMRA	644	648#					
XMTMRB	646	649	654#				
XMTMRQ	540	642#	1256	1421	1516	1655	
XMTREQ	149#						
XMVRER	1542	1546	1550	1554	1564#		
XRCMMN	1072	1083	1085#				
XRCVR1	1091	1096#					
XRCVR2	1098	1105#					
XRCVR3	1106	1114#					
XRCVR4	1094	1103	1112	1115	1124#		
XRCVRD	761	1082#					
XRCVRM	599	1071#	1277	1442	1537	1676	1760
XRNVEF	1114	1126#					
\$RETF	119						
\$RETI F	119						
\$RETI Y	119						
\$RETT	119						
..0001	886	886#					
..0002	900	900#					
..0003	993	993#					
..0004	1047	1047#					
..0005	1796	1796#					
..0006	1841	1841#					
..0007	1872	1872#					
..0010	1903	1903#					
..0011	1962	1962#					
..0012	2015	2015#					

..0013	2053	2053#													
..0014	2088	2088#													
..0015	2123	2123#													
..0016	2153	2153#													
..0017	2182	2182#													
..0020	2214	2214#													
..0021	2249	2249#													
..0022	2278	2278#													
..0023	2305	2305#													
..MX1	483#	483	484	950#	950	951	964#	964	965	978#	978	979			
..MX2	483#	483	484	950#	950	951	964#	964	965	978#	978	979			
..TSA1	885#	885	899#	899	992#	992	1046#	1046	1795#	1795	1840#	1840	1871#	1871	
	1902#	1902	1961#	1961	2014#	2014	2052#	2052	2087#	2087	2122#	2122	2152#	2152	
.A13	2181#	2181	2213#	2213	2248#	2248	2277#	2277	2304#	2304	2014	2052	2087	2122	2152
	885	899	992	1046	1795	1840	1871	1902	1961	2014	2052	2087	2122	2152	
.A14	2181	2213	2248	2277	2304	1840	1871	1902	1961	2014	2052	2087	2122	2152	
	885	899	992	1046	1795	1840	1871	1902	1961	2014	2052	2087	2122	2152	
.A15	2181	2213	2248	2277	2304	1840	1871	1902	1961	2014	2052	2087	2122	2152	
	885	899	992	1046	1795	1840	1871	1902	1961	2014	2052	2087	2122	2152	
.A16	2181	2213	2248	2277	2304	1840	1871	1902	1961	2014	2052	2087	2122	2152	
	885	899	992	1046	1795	1840	1871	1902	1961	2014	2052	2087	2122	2152	
.ACB	2181	2213	2248	2277	2304	1840	1871	1902	1961	2014	2052	2087	2122	2152	
	885#	885	886#	886	899#	899	900#	900	992#	992	993#	993	1046#	1046	
	1047#	1047	1795#	1795	1796#	1796	1840#	1840	1841#	1841	1871#	1871	1872#	1872	
	1902#	1902	1903#	1903	1961#	1961	1962#	1962	2014#	2014	2015#	2015	2052#	2052	
	2053#	2053	2087#	2087	2088#	2088	2122#	2122	2123#	2123	2152#	2152	2153#	2153	
	2181#	2181	2182#	2182	2213#	2213	2214#	2214	2248#	2248	2249#	2249	2277#	2277	
	2278#	2278	2304#	2304	2305#	2305									
.ACM	885#	885	886	899#	899	900	992#	992	993	1046#	1046	1047	1795#	1795	
	1796	1840#	1840	1841	1871#	1871	1872	1902#	1902	1903	1961#	1961	1962	2014#	
	2014	2015	2052#	2052	2053	2087#	2087	2088	2122#	2122	2123	2152#	2152	2153	
	2181#	2181	2182	2213#	2213	2214	2248#	2248	2249	2277#	2277	2278	2304#	2304	
	2305														
.LBCON	1965	1993	2001												
.LBCRM	2155	2168	2171												
.LBCSP	2184	2197	2204												
.LBDIS	2055	2072	2075												
.LBEVT	2017	2034	2042												
.LBGDE	2290	2292	2295												
.LBGRM	2125	2138	2141												
.LBRCD	1904	1930	1950	2459											
.LBRMG	2251	2263	2268												
.LBSPG	2091	2107	2112												
.LBSTC	2216	2233	2237												
.NVR	885#	885	886	886#	899#	899	900	900#	992#	992	993	993#	1046#	1046	
	1047	1047#	1795#	1795	1796#	1796	1840#	1840	1841#	1841#	1871#	1871	1872#	1872#	
	1902#	1902	1903#	1903#	1961#	1961	1962#	1962#	2014#	2014	2015#	2015#	2052#	2052	
	2053	2053#	2087#	2087	2088#	2088	2122#	2122	2123#	2123#	2152#	2152	2153#	2153#	
	2181#	2181	2182	2182#	2213#	2213	2214	2214#	2248#	2248	2249#	2249#	2277#	2277	
	2278	2278#	2304#	2304	2305	2305#									
.POPJ	119														
.PSECT	483	484	885	899	950	951	964	965	978	979	992	1046	1795	1840	
	1871	1902	1961	2014	2052	2087	2122	2152	2181	2213	2248	2277	2304		

.RETF	119
.RETT	119
.SECRA	491
.SQADB	2158
.SQADC	1978
.SQAFB	2128
.SQAMC	1976
.SQAPT	2096
.SQBDN	2187
.SQBNA	1195
.SQCDT	1974
.SQCID	1051
.SQDIS	2058
.SQDPN	1970
.SQDST	1359
.SQESB	1053
.SQEVT	1055
.SJFLG	2094
.SQFST	954
.SQLEN	1905
.SQLPT	2097
.SQOSB	1907
.SQRCI	1986
.SQSPN	1968
.SQSYS	1972
.SQXFL	2316
.SSCON	1962
.SSCRM	2159
.SSCSP	2188
.SSDIS	2059
.SSEVT	2020
.SSGDE	2283
.SSMAP	2318
.SSQRM	2129
.SSRCD	1908
.SSRMG	2254
.SSSMG	2098
.SSSTS	2219

ERROR1	432 758 1387 1634	455 764 1424 1658	465 767 1427 1661	474 770 1445 1679	542 773 1449 1683	544 798 1453 1687	572 1235 1457 1691	580 1259 1480 1712	602 1262 1519 1747	606 1280 1521 1763	610 1284 1540 1767	614 1288 1544 1771	694 1292 1548 1775	697 1318 1552	SEQ 0654
GLOB	119														
ITEXT	227	228													
JUMPF	119														
JUMPT	119														
LSTOF.	119 1035 2214	168 1040 2249	205 1047 2278	218 1796 2305	227 1841	228 1872	398 1903	459 1962	886 2015	900 2053	936 2088	943 2123	944 2153	993 2182	
LSTON.	886 2182	900 2214	993 2249	1047 2278	1796 2305	1841	1872	1903	1962	2015	2053	2088	2123	2153	
MOVX	483	950	964	978											
NAME	114														
PJRST	119														
PROLOG	118														
\$CALL	165 248 432 509 571 642 712 771 952 1039 1254 1313 1428 1486 1553 1661 1752 1923	167 251 443 520 572 643 716 774 953 1049 1256 1314 1431 1508 1556 1664 1755 1983	172 262 445 522 577 648 729 789 966 1071 1259 1319 1432 1514 1574 1665 1757 2025	173 265 447 525 578 661 732 794 967 1082 1263 1328 1435 1516 1582 1668 1760 2064	179 278 453 538 580 665 734 798 980 1105 1266 1332 1439 1519 1585 1676 1764 2100	186 284 455 540 584 667 737 813 981 1114 1267 1335 1442 1522 1588 1680 1768 2131	197 286 458 542 593 669 740 818 994 1134 1270 1337 1446 1525 1589 1684 1772 2161	204 291 463 545 595 688 746 832 1000 1157 1277 1339 1450 1527 1621 1688 1776 2190	216 299 465 551 599 690 747 847 1002 1164 1281 1393 1454 1530 1624 1692 1779 2224	217 346 474 552 603 694 751 849 1009 1167 1285 1393 1458 1534 1634 1706 1783 2256	233 397 485 560 607 698 758 887 1017 1170 1289 1413 1461 1537 1649 1707 1809 2285	234 410 493 563 611 704 761 935 1019 1197 1293 1419 1475 1541 1653 1713 1824 2320	238 415 504 565 615 705 764 942 1026 1203 1297 1421 1476 1545 1655 1730 1827 2323	239 418 506 568 629 708 767 943 1034 1235 1301 1424 1481 1549 1658 1747 1913	
\$RET	119# 479 695 895 1144 1615 2134	119 487 699 906 1184 1630 2164	207 495 702 928 1233 1698 2193	221 505 706 936 1304 1744 2229	229 531 722 944 1346 1787 2259	300 543 756 958 1389 1830 2288	337 546 759 972 1425 1861 2327	373 549 790 986 1429 1892	402 553 795 1031 1465 1917	411 589 826 1035 1477 1924	444 597 835 1040 1520 1989	456 630 865 1050 1523 2030	466 654 872 1060 1560 2068	475 678 879 1124 1609 2103	
\$RETF	119#														
\$RETIF	119#														
\$RETI	119#														
\$RETT	119#														
\$SAVE	885 2181	899 2213	992 2248	1046 2277	1795 2304	1840	1871	1902	1961	2014	2052	2087	2122	2152	
\$TEXT	167	204	217	397	458	935	942	943	1033	1037					
..POP	886 2182	900 2214	993 2249	1047 2278	1796 2305	1841	1872	1903	1962	2015	2053	2088	2123	2153	
..PUSH	886 2182	900 2214	993 2249	1047 2278	1796 2305	1841	1872	1903	1962	2015	2053	2088	2123	2153	

